TAMPERE UNIVERSITY OF TECHNOLOGY

Faculty of Computing and Electrical Engineering
Department of Electronics and Communications Engineering

KANTHI MEENAL PALANIAPPAN

SYSTEM INTEGRATION AND VERIFICATION OF GNSS BASEBAND PROCESSOR

Master of Science Thesis

Subject approved in the Faculty Council meeting on 20th of February 2010

Examiners: **Prof. Jari Nurmi**
Dr.Heikki Hurskainen, D.Sc.(Tech)

Preface

This thesis work was done at the Department of Computer Systems, Tampere University of Technology as a part of European Union projects on GNSS Receiver Reference Design.

I owe my deepest gratitude to my scientific supervisor Prof. Jari Nurmi, who introduced me to this research topic and motivated me. His constant guidance and supervision has been highly important for this thesis work. I thank Dr.Heikki Hurskainen for his expert inputs to this thesis.

Special thanks to my colleagues Fabio Garzia, Zhongqi Liu, Tommi Paakki and Guanchao Xu for the numerous fruitful discussions during the thesis.

I wish to thank all the people in the Department of Computer Systems for the motivating and enjoyable working atmosphere. I would like to thank my husband Prasanna for his love and never-ending support during this thesis work. This preface would be highly incomplete without thanking my kids who have always been much co-operative all through-out this thesis.

Tampere,

5th February 2014.

Kanthi Meenal Palaniappan

Maijalankatu 9 F 22

33720 Tampere

Contents

List of figures

List of tables

Abstract

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Program in Information Technology
Major: Digital and Computer Electronics
PALANIAPPAN, KANTHI MEENAL: System integration and verification of GNSS baseband processor
Thesis Examiners: Professor Jari Nurmi
Dr. Heikki Hurskainen, D.Sc.(Tech)
February 2014

Keywords: GNSS, baseband processing, control processing, interface, verification

Satellite navigation, in the last three decades, has seen an evolution bringing up entirely new systems (Galileo) and modernization of existing systems (Global Positioning System). These systems have now changed the environment of the receiver design resulting in the development of Global Navigation Satellite System (GNSS) with new signal processing algorithms. GNSS receiver receives the signals from a GNSS satellite constellation, digitally processes them and provides position, velocity and time. Hardware GNSS receivers have good efficiency, good computational load and low power consumption. Such a hardware GNSS receiver is presented here.

GNSS Receiver Reference Design is a fully functional L1 only GNSS receiver design. The main objective for this design is to make fully open access architecture (HW + SW) available to industry partners and researchers for development of GNSS and GNSS-enhanced devices, for investigating current GNSS receivers and receiver algorithms and upcoming GNSS receiver standards.

Baseband processing generates pre-processed data from received signals. It comprises digital signal processing executed by custom hardware (baseband system) and control processing implemented by a soft-core processor (COFFEE RISC core). The baseband system component performs acquisition and tracking of 6 channels. It currently

provides only GPS coarse/acquisition (C/A) code. It is implemented by Field Programmable Gate Array (FPGA) logic, supported by hardware macros.

The baseband system and the processor are to be integrated efficiently to manage the receiver activity. The integration is achieved by designing an interface that is compatible with the standard bus architecture. The interface is a shared system bus that contains a register database. The interface is implemented in RTL and verified in functional simulations.

In this thesis, another objective of verifying the baseband system is achieved by targeting the maximum code coverage. The results show that this improves the quality of verification and provides good confidence in the design. The coverage numbers prove that the verification is extensive, close to 100%.

Finally, synthesis is also needed for verifying the design implementation on gate level. Since the baseband system included many of Xilinx based models, both the subsystems are synthesized on Xilinx Virtex-II Pro platform. The synthesis results provide information on the on-chip area consumption.

List of abbreviations

A/D	Analog-to-Digital
ADC	Analog to Digital Converter
BB	BaseBand
BOC	Binary Offset Carrier
C/A	Coarse/Acquisition
CCB	Core Configuration Block
CGCS	China Geodetic Coordinate System
CLA	Carry Look Ahead
CS	Commercial Services
D/A	Digital-to-Analog
DSP	Digital Signal Processing
FPGA	Field Programmable Gate Array
FSM	Finite State Machine
GCLK	Global CLocK
GLONASS	GLObal NAvigation Satellite System
GNSS	Global Navigation Satellite System
GPR	General Purpose Register
GPS	Global Positioning System
GTRF	Galileo Terrestrial Reference Frame
I/O	Input / Output
IOB	I/O Block
IRNSS	Indian Regional Navigation Satellite System
LUT	Look Up Table
MULT	MULTiplier
NCO	Numerically Controlled Oscillator
NMEA	National Marine Electronics Association
OS	Open Services
PCB	Peripheral Configuration Block

PE-90	Parameters of the Earth 1990
PRS	Public Regulated Services
QPSK	Quadrature Phase Shift Keying
RAM	Random Access Memory
RF	Radio Frequency
RISC	Reduced Instruction Set Computing
RTK	Real Time Kinematic
SBAS	Satellite Based Augmentation System
SoL	Safety-of-Life
SPR	Special Purpose Register
WGS 84	World Geodetic System of 1984

1 *Introduction*

1.1 INTRODUCTION

Since many years now, personal positioning has become an everyday solution in location based services. This brings a looming number of consumer applications affecting many aspects of human life.

Of many satellite based systems, currently GPS is the most widely used positioning system found in many things like cell phones, wrist watches, ATMs and shipping containers. Some of the well-known GPS applications are listed below.

- Agriculture: GPS is used in some of the precision agriculture operations like yield monitoring, compaction profile sensing, tree planting site-specific fumigant application, RTK GPS based plant mapping, precision weed management system and robotic applications (Perez-Ruiz & Upadhyaya, 2012).
- Aviation: GPS increases the safety and efficiency of aviation operations.
- Disaster relief: GPS enhances the capability for predicting flood and monitoring seismic events. It delivers relief to disaster areas in a timely way by mapping the location of victims.
- Science: GPS helps in analysis of environmental concerns, efficient tracking of environmental disasters, monitoring and preservation of endangered species (Mendizabal et. al., 2009).
- Maritime: GPS increases the efficiency, safety and optimisation of all marine applications like recreational boats, commercial vessels, and unregulated and safety of life at sea (Mendizabal et. al., 2009).
- Public safety: GPS gives the positional information in case of emergency.

- Transport: GPS improves the safety of trains by preventing collisions and derailments. It increases the capacity and efficiency for all surface transportation system users.
- Surveying and mapping: GPS provides faster and highly accurate data in surveying coasts and waterways.
- Time reference: GPS technology is used wherever there is a need for precise synchronization like electronic banking, e-commerce, the stock exchange, and quality assurance systems and services, wireless telecommunication network management, or power plant and network monitoring, and so on (Mendizabal et. al., 2009).

Thus, GPS has become a vital commodity that enhances the quality of life. Many new and modern positioning applications, that are created every day, are the main driving strengths for faster deployment of accurate, feasible, reliable and cost-effective positioning technologies. Thus, the five important performance measurements are presented here below.

Table 1.1 Performance measurements of a positioning technology

Performance measurement	**Detail**
Accuracy	This gives a measure of how close is the measured location to the actual location.
Reliability	This gives a measure of how consistently GPS error levels could be maintained below a specified threshold.
Availability	This gives a measure of how GPS signals are available without any blockage in all of the remote, rural, suburban, urban, indoor and underground areas.
Latency	This gives a measure of how long a time is taken to get the first location measurement.
Applicability	This gives a measure of the practical requirements and restrictions in using the positioning technology.

Many researches are being conducted around the world in positioning technologies, systems and solutions to meet the demands specified above. These demands bring a lot challenges to the satellite based positioning technology like GNSS.

1.2 TARGETS AND OUTLINE OF THE THESIS

The GNSS Receiver Reference Design aims to be an open source hardware receiver. The motivation of this thesis lies in here. Designing the baseband processor system contributes largely towards it by filling the gap between Radio and Navigation components. The other major contribution is in realizing a real-time GNSS receiver in the future. Having the two subsystems 'baseband' and 'processor' ready, the primary focus is on designing the Interface. The secondary focus is on exhaustive functional verification of the baseband processor system as GNSS receiver is a critical application of satellite navigation.

This thesis can be divided in to two parts. The first part, chapters two, three and four provide the background information that is needed for the reader before proceeding with the author's work. The second part, chapters five, six and seven cover the actual interface design, verification and synthesis of baseband processor system.

Chapter 2 is an introductory part providing some knowledge about GNSS, GNSS signals, GNSS receiver and GNSS Receiver Reference Design.

Chapter 3 is a description of the baseband subsystem which is a result of the study conducted about the baseband subsystem functions and implementation.

Chapter 4 is a similar study conducted for the processor subsystem. It includes various details like the philosophy, features and interface specification of the COFFEE core that was used to construct the processor subsystem.

Chapter 5 covers one of the main tasks of this thesis. The interface is needed to integrate the two subsystems. The interface design has followed a five-step process which is stated in detail here.

Chapter 6 contains the other main task. The baseband subsystem is verified completely with a proper verification strategy and planning. The results of the detailed verification are shown here.

In Chapter 7 the synthesis of both baseband subsystem and processor subsystem are performed onto Xilinx Virtex-II Pro FPGA platform. The results are given here.

Chapter 8 is the conclusion part that also presents the possible future work.

2 *GNSS Receiver Reference Design*

2.1 GNSS

GNSS commonly addresses all the satellite navigation systems that has a global coverage to provide precise positioning. It is an evolution in the satellite navigation. During the recent years, the GNSS prospect has changed greatly. In particular, GNSS solutions are demanded such that they offer high precision, reliability, configurability, authentication, service guarantees, security, and anti-jamming. Some of the major players in GNSS are Global Positioning System (GPS), GLObal NAvigation Satellite System (GLONASS), Galileo, and Compass (Beidou-2). A comparison of these systems is given in Table 2.1 below.

Table 2.1 A comparison of GNSS systems

	GPS	**GLONASS**	**Galileo**	**Compass**
Country	USA	Russia	EU	China
Current state of development	Operational; Being modernized	Being restored to full operation	Initial deployment phase	Expanding the regional system to the global system
Orbits	6	3	3	6 (Grace, 2007)
Orbit period **(sidereal days)**	~11h 58m (1/2)	~11h 15m (8/17)	~14h 07m (10/17)	~12h 55m (7/13)
Satellites	30	24	30	30 + 5 Geo
Max. Range Rate (Diggelen, 2009)	+/- 800 m/s	+/- 900 m/s	+/- 650 m/s	+/- 700 m/s
Coordinate systems (Diggelen, 2009)	WGS 84	PE-90	GTRF	CGCS2000
Inclination (Diggelen, 2009)	55 deg	64.8 deg	55.9 deg	55 deg

The total number of satellites planned in GNSS is 137 which include satellites of GPS, GLONASS, Galileo, Compass, Indian Regional Navigational Satellite System (IRNSS) and Satellite-based Augmentation System (SBAS). As Diggelen mentions accuracy, especially in urban environments, will be improved dramatically by the large number of satellites available when most or all of the planned GNSS constellations are complete (Diggelen, 2009). Figure 2.1 shows the complete constellation of all the planned satellites.

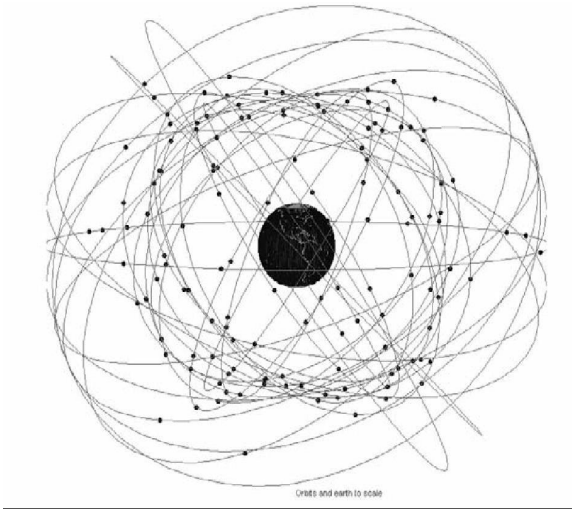

Figure 2.1 The complete constellation of all planned GNSS (Diggelen, 2009)

2.2 GNSS SIGNALS

Table 2.2 below shows the different signals of GPS, GLONASS, Galileo and Compass that consist of different frequencies.

Table 2.2 GNSS frequencies and signals

	Frequencies	**Signals**
GPS (including modernisation)	All satellites transmit at the same frequencies. L1 carrier - 1575.42 MHz L2 carrier - 1227.60 MHz L5 carrier - 1176.45 MHz	C/A (coarse acquisition) P (precision), M (military) L1-C/A, L1-P, L1-M L2-C/A, L2-P, L2-M L5-SoL
GLONASS	All satellites transmit at slightly different carrier frequencies. L1 ~1.6 GHz, L2 ~1.2 GHz	L1-C/A, L1-P L2-C/A (Revnivykh, 2009), L2-P

Galileo	All satellites transmit in the same frequency bands. E5a, E5b – 1164 to 1215 MHz E6 – 1215 to 1300 MHz E2-L1-E1 – 1559 to 1592 MHz	10 different signals. Six on E5a, E5b and L1 for OS and SoL. Two on E6 for CS. One on E6 for PRS. Finally, one on E2-L1-E1 for PRS (Guenter W. Hein, 2002).
Compass	All satellites broadcast in three frequency bands (Chong, 2009). B1 - 1559.052 to1591.788 MHz B2 - 1166.22 to 1217.37 MHz B3 - 1250.618 to 1286.423 MHz	8 different signals. 4 QPSK modulated signals and 4 BOC modulated signals on B1, B2 and B3 (Chong, 2009).

As seen above, all the systems have some frequency bands common with the other systems. Thus, GNSS systems are compatible such that they can be used separately or together without causing unacceptable interference and/or other harm to an individual system and/or service. They are interoperable such that the services they provide can be used together to provide better capabilities at the user level than would be achieved by relying solely on the open signals of one system (Revnivykh, 2009). Figure 2.2 shows the complete GNSS spectrum for the frequencies and signals planned for GPS, GLONASS, Galileo and Compass.

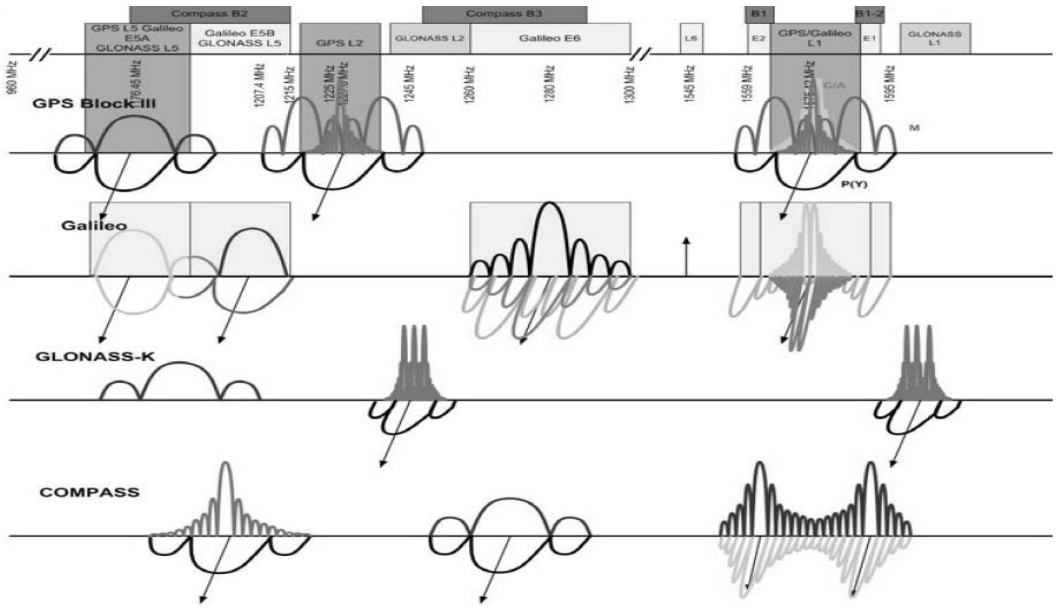

Figure2.2 Complete GNSS spectrum (Diggelen, 2009)

2.3 GNSS RECEIVER

Having said about the compatibility and interoperability of GNSS systems, a receiver that is capable of receiving such GNSS signals improves all aspects of satellite navigation – availability, reliability, stability, etc. With many satellites (60+), performance is improved in urban canyons and also in other locations with restricted visibility.

To calculate a user position, firstly, a GNSS receiver must find the satellites and acquire their signals and then decode the data to find the position of the satellites. Then, the receiver has to compute the user position. The basic functions of a GNSS receiver are stated below (Tsui, 2000)

- The signals transmitted from the satellites are received through the antenna.
- The signal is then amplified to a desired amplitude and the frequency is converted to a required output frequency
- Then, the signal is digitized using an Analog-Digital Converter (ADC)
- Acquisition then follows to find the signal of a certain satellite
- Tracking is used to find the phase transition of the navigation data
- Ephemeris data and pseudo ranges are obtained from the navigation data
- The ephemeris data is used to obtain the satellite positions
- The satellite positions and pseudo ranges are then used to calculate the user position.

The challenges in designing a GNSS receiver are:

- the environment which is constantly changing topographically
- multipath
- implementation challenges due to interoperability
- radio front end design for the increasing amount of frequency bands
- increased bandwidth of new signals causes interference

2.4 GNSS RECEIVER REFERENCE DESIGN

The GNSS Receiver Reference Design is a university based, fully functional L1 only 6-channel GNSS receiver (Hurskainen, H. et al., 2008). The core of this receiver is contained in one or more highly sophisticated chips that perform all the receiver's tasks, starting with signal processing, followed by positioning and ending at application processing. This reference design is ideally suited for research and development of

GNSS receivers. The receiver features a Xilinx Virtex-II Pro FPGA with embedded memory. It covers a wide range of features to allow a number of different GNSS related applications to be researched and developed. The main objectives of GNSS receiver reference design are:

1. To develop a platform for investigating current GNSS receivers and receiver algorithms and upcoming GNSS receiver standards.
2. To make fully open access architecture (HW + SW) available to industry partners and researchers for development of GNSS and GNSS-enhanced devices.

It consists of four major components as shown in Figure 2.3 below.

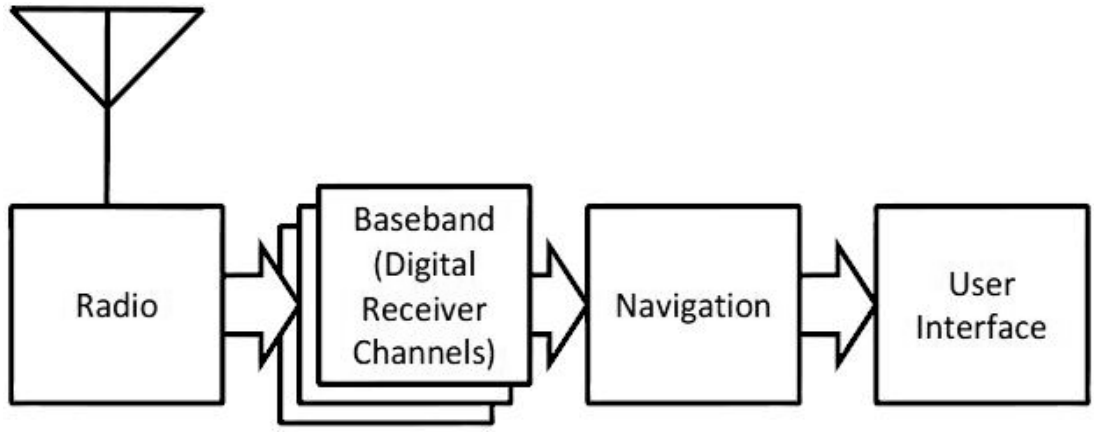

Figure 2.3 Basic block diagram of GNSS Receiver Reference Design (Hurskainen H., 2008)

Radio: This radio part contains many sub-parts to perform specific functions.

i. Antenna – To receive GNSS signals
ii. LNA – To amplify the input spectrum with less noise
iii. Band-pass filter – To pass only GNSS frequencies (1.1 to 1.6 GHz)
iv. Band-stop filter – To remove frequencies that are not of interest (1.3 to 1.54 GHz)
v. ADC – To convert the analog spectrum into digital samples, whose sampling frequency is twice that of the bandwidth.

Since satellite signals are very weak, the development of front-end components requires custom circuit design.

Baseband: This is the main control processing block to manage receiver activity, i.e. processing of digital signals. It generates raw/pre-processed data from the incoming digital signals. This also requires custom hardware design. The next chapter describes more of it.

Navigation: It generates post-processed data, i.e. converts raw data (pseudo ranges and navigation data) into position, velocity and time according to NMEA format, usually executed by a micro-controller running special firmware. From the input data stream, firstly, navigation data is constructed and time stamp is generated. Then, data integrity is checked and ephemeris data is collected. User location estimates are made which contains random error. Kalman filtering is done to provide smoother user location. Lastly, the user location and time information are converted to NMEA format.

User Interface: It interacts with the application code and user, providing user control and displaying information. It generally displays current information – the user's position, time and date. Firstly, it collects data associated with position information. Then, it provides the data and mapping displays. It also provides navigational functions to permit a user to navigate to a desired location with audio and visual feedback.

3 *Baseband Subsystem*

Baseband signal processing is achieved using a combination of a dedicated hardware and a processor. The dedicated hardware, 'Baseband Subsystem', is discussed in detail here.

3.1 BASEBAND SUBSYSTEM FUNCTIONS

The two main functions of the baseband subsystem are Acquisition and Tracking. This baseband is capable of tracking signals on L1 band. The fundamental tasks are measurement of Doppler frequency and code delay, which is further processed in software to find the pseudo range, and demodulation of the navigation data.

3.1.1 Acquisition

The main role of the acquisition process is to acquire the satellite signal. In order to achieve this, code phase search and also frequency search is done for each of the satellite signal. This is because the relative motion of the satellites with respect to the receiver brings in some Doppler frequency on the transmitted signal which is roughly +/-10 kHz (Berger, 1996). Thus, this process requires replication of both code and carrier.

- The range dimension is associated with the replica code (E.D. Kaplan, 2006) where the code phase is searched in ½ chip increments. The replica must be well aligned in time to the PRN code in the received satellite signal. This PRN code is unique to each satellite. Using this code, the receiver identifies each satellite

and, consequently, calculates the time of travel of the signal from the satellite to the receiver.

- The frequency dimension is associated with the Doppler frequency shift (E.D. Kaplan, 2006). The frequency of the carrier replica must match the frequency of the carrier in the received signal. The frequency is searched through in 500 Hz increments.

After the signal has been found, rough estimates of Doppler frequency and code phase values are made. Acquisition also refines carrier search results if it is needed for the tracking. A simple acquisition process is shown in Figure 3.1 below.

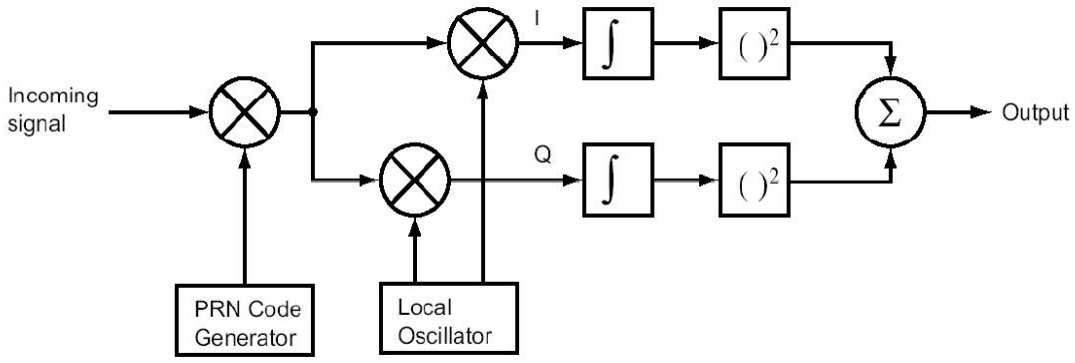

Figure 3.1 Acquisition process (Plausinaitis, 2009)

3.1.2 Tracking

The aim of tracking is to follow the signal and obtain the information of the navigation data.

The method of tracking involves a narrow-band filter that is built around the input signal which is then followed. Keeping the centre frequency of the filter fixed, a locally generated signal follows the frequency of the input signal. The phases are compared through a phase comparator. The output from the phase comparator passes through a narrow-band filter (Tsui, 2000).

To track a signal, two tracking loops are required.

- One loop is to track the carrier frequency, referred to as the carrier loop
- The other loop is to track the C/A code, referred to as the code loop.

Tracking is implemented such that both these loops use a Numerically Controlled Oscillator (NCO). The carrier NCO with discrete sine and cosine mapping functions

synthesize the replica carrier signals. This replica is mixed with the incoming signal to remove the residual carrier (plus Doppler) and to produce in-phase (I) and quadrature (Q) data samples. These samples are mixed with early, prompt and late versions of the PRN code and then accumulated to form the associated correlation values. If the replica code is aligned, no error is generated by feedback loops. Otherwise, the feedback loops sense the error. This error is then filtered so as to apply to the code NCO where the output frequency is corrected to match the replica code phase with the satellite signal code phase. Here, Delay Locked Loop (DLL) is used for tracking PRN code phase whereas Frequency Locked Loop (FLL) is used for tracking signal's carrier frequency and Phase Locked Loop (PLL) is used for tracking signal's carrier phase. Figure 3.2 below shows the tracking unit of a GPS receiver.

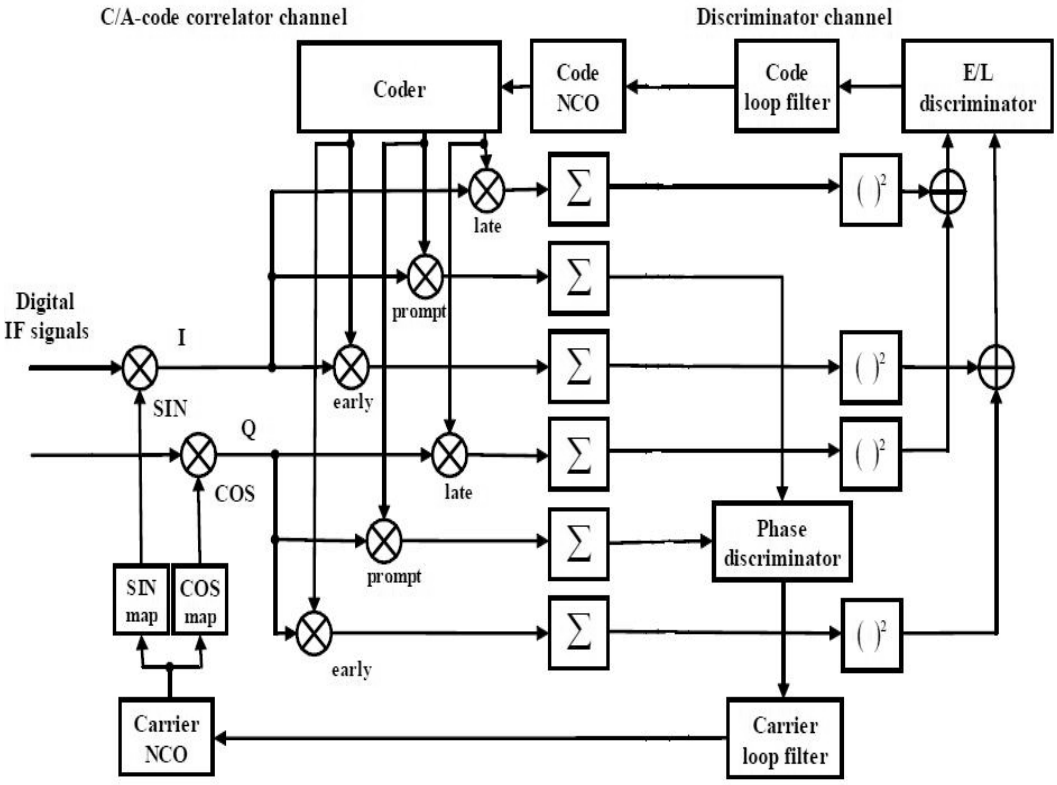

Figure 3.2 Tracking unit of a GPS receiver (Dierendonck, 1999)

3.2 BASEBAND SUBSYSTEM IMPLEMENTATION

The actual implementation of the baseband subsystem consists of one acquisition block and 6 tracking channels. After acquiring a satellite signal, the tracking process is started in one of the tracking channels while the acquisition block continues to search for the next satellite.

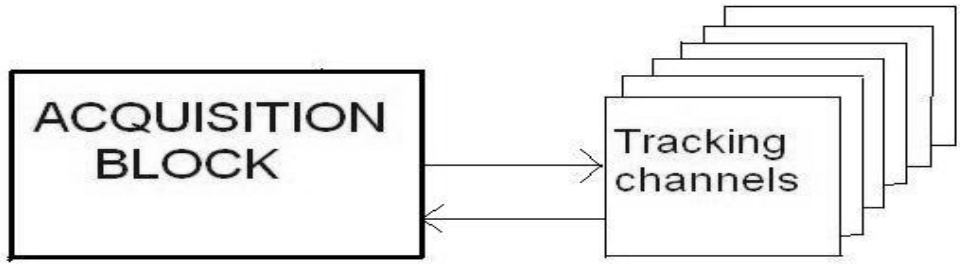

Figure 3.3 Acquisition and Tracking channels

The acquisition block is implemented based on an advanced acquisition algorithm. The basic idea of this advanced algorithm is still based on the serial acquisition algorithm. It has a few trade-off advantages compared to serial and parallel acquisition algorithms.

- 200 times faster (0.2 seconds to acquire one satellite) than conventional serial acquisition algorithm which takes 40 seconds for acquiring one satellite (Liu, 2008).
- Consumes lesser silicon area than parallel acquisition algorithm. It uses three on-chip memories to store PRN codes and I and Q samples. The parallel acquisition algorithm uses 1023 1-bit registers to store PRN codes and 2046 3-bit registers to store I and Q samples (Liu, 2008).

3.2.1 Basic Components

The basic components used in the acquisition block are code generator, mixer, carrier NCO, code NCO, on-chip memories, averaging filter, and CLA adder.

The basic components used in the tracking block are correlator, accumulator, code shifter, mixer, carrier NCO, code NCO, and code generator.

3.2.2 Component Descriptions

1. ***Mixer***

Aim: To remove the carrier (plus Doppler, if any) from the incoming digital IF signal. This process is termed as carrier wipeoff.

Requirement: The sampled data must have the desired phase relationships with respect to the detected carrier of the desired SV (E.D. Kaplan, 2006).

Implementation: All of the in-view SV signals at the input are complex mixed with the replica carrier. This produces in-phase (I) and quadrature-phase (Q) sampled data.

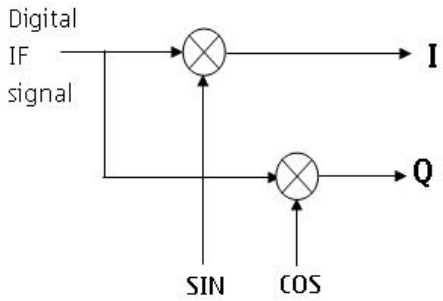

Figure 3.4 Mixer

2. ***Carrier NCO***

Aim: To synthesize the replica carrier (plus Doppler).

Method: To do this, the sine and cosine mapping functions are needed to be created as seen in the above Mixer. The NCO produces a staircase function with a period same as that of the replica carrier plus Doppler. The sine and cosine mapping functions convert each of the discrete amplitude of the staircase function to the corresponding discrete amplitude of the respective sine and cosine functions.

Implementation: It has an internal 32-bit modulo counter to create sine and cosine waves.

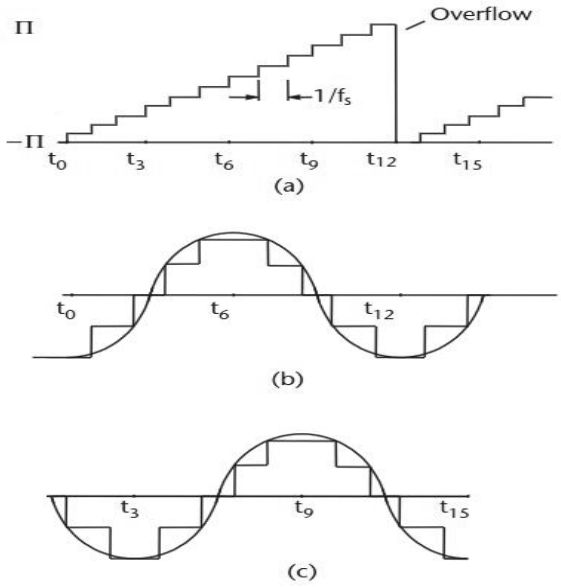

Figure 3.5 (a) NCO output (b) COS map output (c) SIN map output (E.D. Kaplan, 2006)

3. Code NCO

Aim: to control the frequency of the locally generated C/A code sequence (PRN chip rate).

Requirement: To keep the carrier wipe off complexity minimal.

Method: The code wipe off function is implemented after the carrier wipe off function.

Implementation: It has two counters to produce two clocks

i. The code generator clock – consists of the nominal spreading code chip rate (plus code Doppler).
ii. The 2-bit shift register clock - It is twice that of the code generator.

The NCO clock rate should be much higher than that of shift register clock.

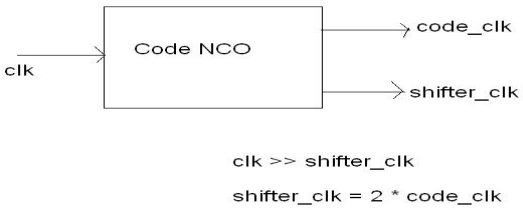

Figure 3.6 Code NCO

4. Averaging filter

Aim: To reduce the noise in the incoming signal after the carrier wipe off.

Requirement: To make the peak value prominent from the noise levels, otherwise, this will be difficult to acquire satellites after correlation.

Method: This filter averages 16 input samples to produce one output sample thus removing high frequency component in the signal.

Implementation: 16 successive samples are added together which is then converted to a decision value like shown below.

If summation result is	Then, decision value is
Greater than or equal to 32	3
Less than 32, greater than or equal to 24	2
Less than 24, greater than or equal to 0	1
Less than 0, greater than or equal to -24	-1
Less than -24, greater than or equal to -32	-2
Less than -32	-3

5. Code generator

The advanced acquisition algorithm of the acquisition block needs this serial code generator.

Aim: To produce PRN code sequentially at the rate of 1.023 MHz clock that is coming from the Code NCO block.

Requirement: The PRN code sequence must have the same chip phase as that of the incoming GPS signal.

Method: At the start of the acquisition, the contents of all the internal registers are cleared and the PRN code is produced by searching all cell combinations for the satellite number given by the software.

Implementation: It is capable of producing all 37 GPS PRN codes. It generates the controls for on-chip memories – memory addresses for writing PRN code and incoming signal samples, write enable for PRN code and write enable for signal data.

6. On-chip memories

The advanced acquisition algorithm of the acquisition block also needs these on-chip memories.

Requirement: Dual-port memory in two different clock domains.

Aim: Three memories to store PRN code, I and Q samples. One memory is to synchronize the control signals and correlation results between the two clock domains.

Implementation: In the 50 MHz domain, PRN code, I and Q samples are written into memories at 1.023 MHz chip rate. In the 200 MHz domain, the same PRN code, I and Q samples are read by the correlation block at the rate of 200MHz.

7. CLA adder

Aim: To increase the acquisition algorithm speed and to achieve the maximum clock frequency of 200MHz in the correlation process.

Requirement: To avoid waiting for the carries to ripple through each successive bit of the adder.

Method: CLA adder is used in the correlation process instead of the conventional adder.

Implementation: 13-bit CLA adder advances C_{in} directly to C_{out} through the AND/OR gate in each bit until the last.

8. Correlators & Accumulators

I and Q samples are correlated separately with the PRN code. Thus, 6 instances of this correlator are used as shown in Figure 3.7 below.

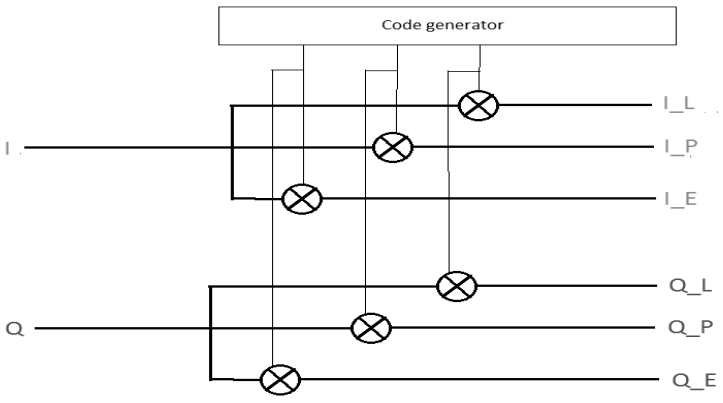

Figure 3.7 Correlators

If code is 1, then, $d_{out} = d_{in}$
Else, $d_{out} = -d_{in}$

The sign inversion is done with two's complement.

These I_L, I_P, I_E, Q_L, Q_P and Q_E data values are accumulated separately using 6 accumulators. The accumulated values are then dumped out when needed.

9. Code shifter

Aim: To produce early, prompt and late replicas of the PRN code.

Method: A block diagram of the code shifter is shown in Figure 3.8 below. The frequency of shift_clk will determine the phase difference between the input and the outputs.

Chip phase difference = chipping frequency / shift_clk

If the frequency of the shift_clk is double the chipping frequency, the phase difference between different outputs will be 0.5 chip.

Implementation: The replication is done using shift registers.

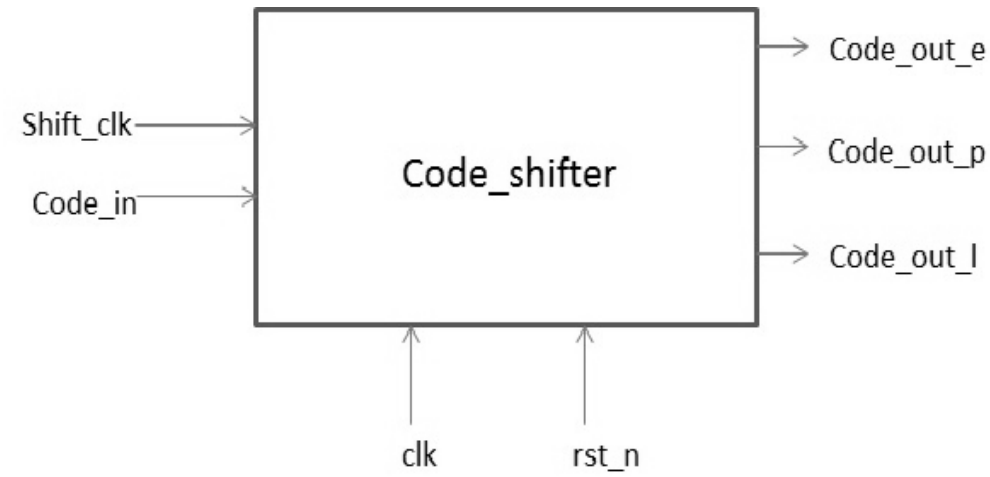

Figure 3.8 Code shifter

4 *Processor Subsystem*

4.1 COFFEE RISC CORE

The processor sub-system that is selected for the Baseband processor is COFFEE RISC core (COFFEE Core User Manual, 2007). This is an open source processor core and a trademark of Tampere University of Technology. The core refers to the hardware block that is responsible for execution of the instruction set of the processor. It has no peripheral components, no buses and no cache memories. COFFEE core, being a RISC type of processor core, is chosen for this baseband application because of the following advantages:

- Hardware is less complex
- Increased performance
- Reduced chip area
- Reduced power consumption
- Technology independent
- Flexible enough for easy instantiation into a larger system

Thus, it is a reasonably powerful processing engine for embedded computing.

4.1.1 COFFEE philosophy

COFFEE core has followed the RISC design philosophy. It has fewer and lesser complex set of instructions. Some of the requirements are 'Pipelining' to achieve one instruction per cycle, to minimize memory traffic (to utilize the pipeline in an optimal way; only load and store instructions access memory), fixed instruction length to keep the decoding of an instruction simple, simplified addressing mode to avoid hardware

penalty from address arithmetic, reduced number and complexity of instructions that would need shorter clock cycles to fit the pipelines better, delayed branching (compiler carefully schedules some meaningful instructions in delay slot(s) after the branch instruction in order to hide the latencies).

4.1.2 COFFEE features

The main features of this core are listed below (Juha Kylliainen, 2003):

- 32-bit architecture
- Balanced pipeline - 6 pipeline stages
- 67 instructions, full-size arithmetic operations including barrel shifter and three versions of multiplication and excluding divider
- 16-bit and 32-bit processor operating modes
- Load-store machine using a vast amount of registers. Two register sets containing 32 registers each including GPRs and SPRs
- A memory mapped register bank, CCB to support operating systems and different configurations. Also, eight condition registers (CRs)
- Coprocessor interface supports up to four coprocessors
- Harvard type of memory interface, separate interfaces for data and instruction memory
- An internal interrupt controller; an external controller can also be used
- PCB block provides easy communication directly with peripheral devices around COFFEE core
- System bus interface; allows sharing of data memory bus with other devices
- Extensive forwarding to simplify the construction of the compiler
- Two independent built-in timers

4.1.3 Interface specification of the COFFEE core

This general purpose processor core can be used as an IP-block, as part of a larger system, in embedded system applications. The functional blocks of the core include interrupt controller, control unit, address checker, instruction memory interface, condition check, instruction decode, condition registers, register file, configuration registers, timers, ALU, data memory interface and coprocessor interface. Figure 4.1 below shows the possible interfaces for the core. The coloured blocks are optional; data and instruction caches can be replaced by interfacing directly to separate local memories respectively.

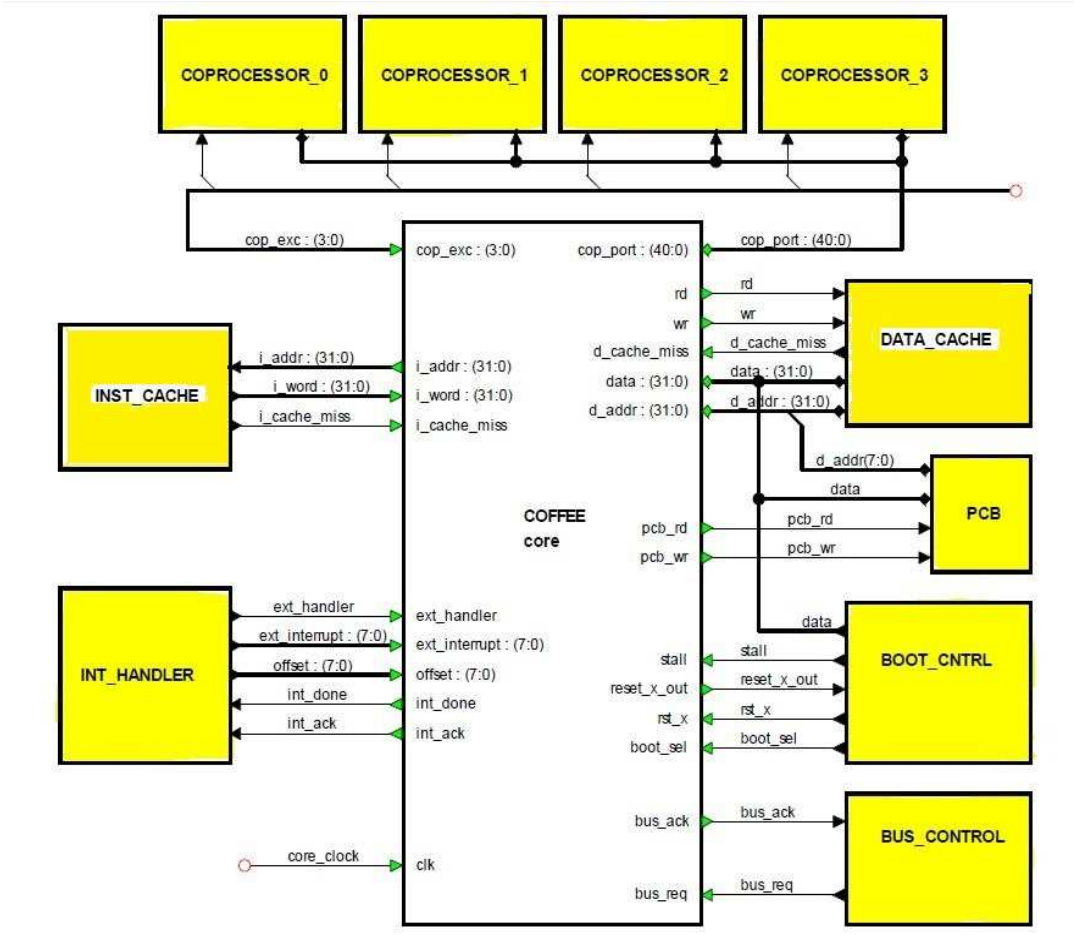

Figure 4.1 Core Interface (COFFEE Core User Manual, 2007)

4.2 PROCESSOR SUBSYSTEM ARCHITECTURE

This subsystem chooses a platform based on standard bus architecture. This architecture is a simple and economical choice. The shared system bus connects all the subsystems of a larger system.

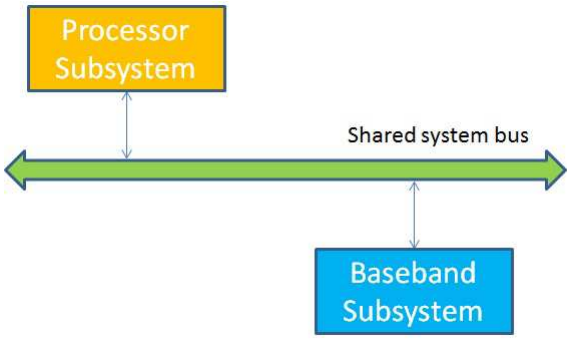

Figure 4.2 Simple bus architecture

The performance of a system is greatly dependent on the characteristics of the interconnect architecture. The interconnect architecture design for a system requires careful consideration of some factors like shown below (Michael J. Flynn, 2011).

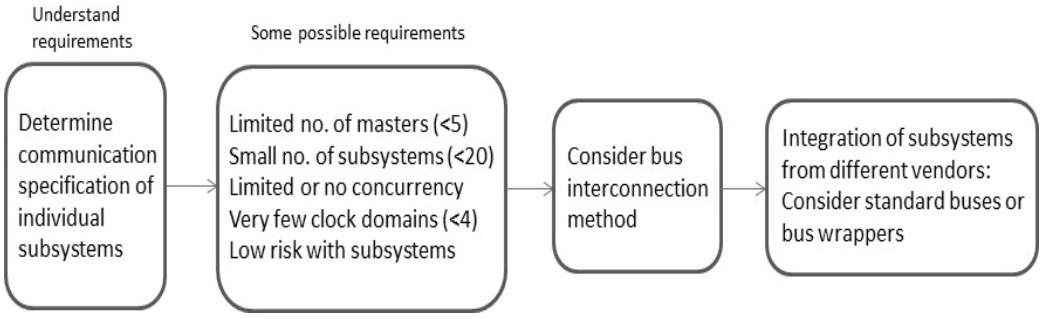

Figure 4.3 An outline for interconnect design (Michael J. Flynn, 2011)

The use of the bus is controlled by some logic so as to avoid conflicts between the subsystems. A subsystem is said to own the bus when it has the exclusive use of the bus. Subsystems that request the ownership (bus_req) and initiate the communication on the bus are called masters. Subsystems that are passive and only respond to requests (bus_ack) are called slaves. The bus master controls the bus paths using specific slave addresses and control signals. It also controls the flow of data signals directly between the master and the slave.

The advantages of this bus architecture are: Bus latency is zero, the silicon cost of a bus is low for small systems, the bus is almost directly compatible with the systems and the concepts are simple and well understood.

4.3 PROCESSOR SUBSYSTEM DESIGN

The basic version includes COFFEE core and some bus control logic as shown in Figure 4.4 below. All the unused inputs such as COP_EXC[3:0], D_CACHE_MISS, EXT_HANDLER, I_CACHE_MISS, EXT_INTERRUPT[7:0], BOOT_SEL and STALL are driven to inactive state, low. All other unused inputs can have any values.

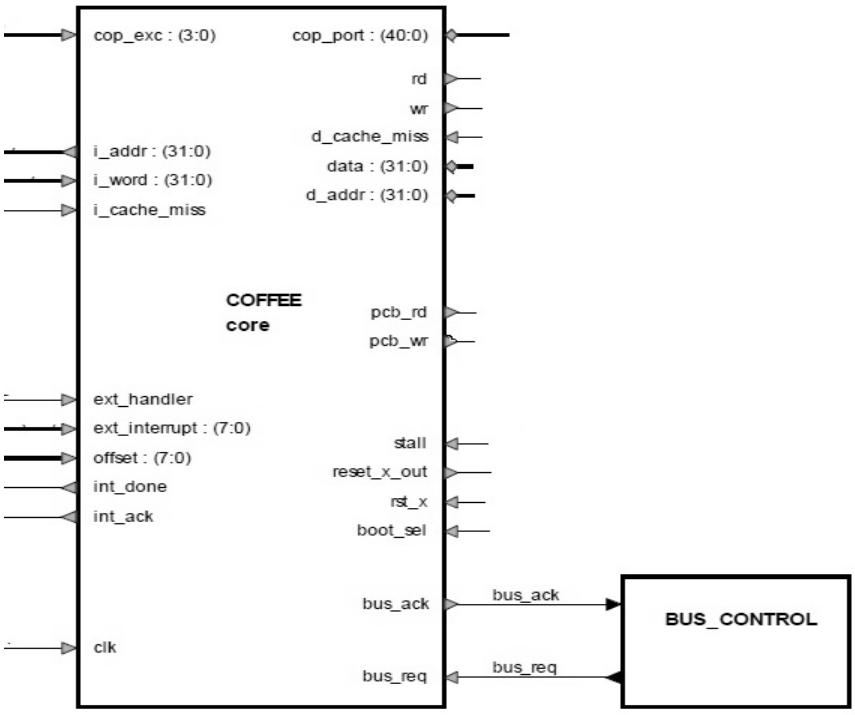

Figure 4.4 Processor subsystem design

5 *Interface Design*

The interface design process has followed the steps (in order) shown below. It is always possible to go to any previous step and proceed from there again.

- Design requirements
- Requirements decomposition
- Functional design
- Functional implementation
- Functional verification

In this chapter, all these steps are described in detail for the interface design.

5.1 INTERFACE – DESIGN REQUIREMENTS

The two design requirements are as under:

1. Baseband subsystem generates raw/pre-processed data from the received signals whereas processor subsystem performs control processing. These two subsystems have to be integrated efficiently so as to manage the receiver activity.
2. The application software to control both acquisition and tracking modules (in baseband subsystem) with data writes to some control registers, and obtain status and measurement data with reads from the channel and system registers.

5.2 INTERFACE – REQUIREMENTS DECOMPOSITION

The above two design requirements are decomposed into small functional requirements. They are:

1. Both the subsystems share the system bus. This implies that the interface must be compatible to the standard bus architecture (bus_req, bus_ack).
2. Based on the wr/rd commands from the processor, the interface must write data into the registers or read data from the registers at the location specified by the processor.
3. It must also generate control signals writeEn and readEn so that the respective operations are preformed correctly.
4. Register database to be created based on the below decisions:
    a. What are the data to be registered from the baseband system and the processor subsystem?
    b. How many registers are needed for each of the data?
    c. What is the register size?

5.3 INTERFACE – FUNCTIONAL DESIGN

The above step, thus, decomposes the interface design into two components – interface and registers. The output of this step is the block diagram of the design and the functional specifications of each of the components and data flow diagrams.

The block diagram of the whole baseband processing system is shown in Figure 5.1 below. The processor subsystem shares the system bus with the baseband subsystem. The baseband subsystem sends bus_req signal to the processor and on receipt of bus_ack, it owns the bus. When there is a write or read operation, the processor puts bus_ack low. It then owns the bus. When there is no write or read operation and there is no request from baseband, the bus is idle with no active accesses; the processor drives last values on the bus.

As shown in Figure 5.1 below, the interface generates the enable signals for write and read operations. There are five register sets for the data coming from the baseband_system and nine register sets for the data going out to the baseband_system.

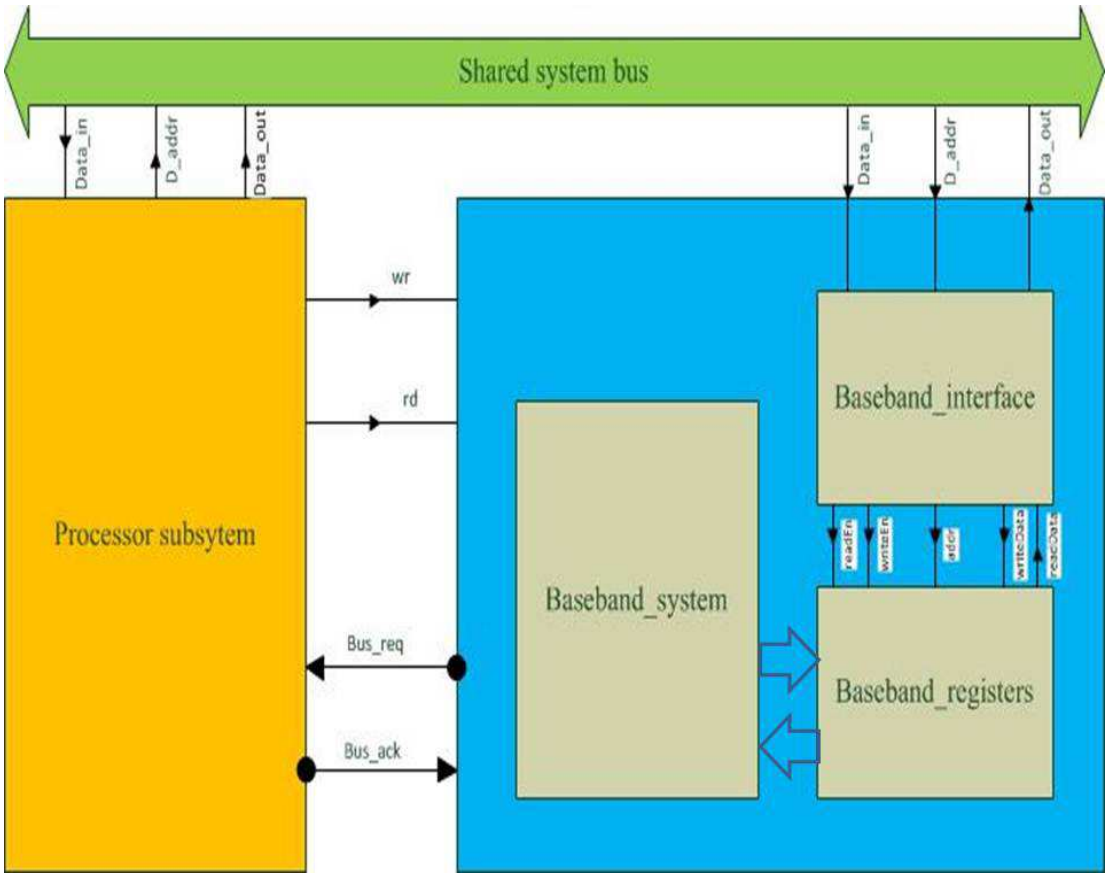

Figure 5.1 Block diagram of baseband processing system

Figures 5.2 and 5.3 show the data flow diagrams for the write and read operations respectively.

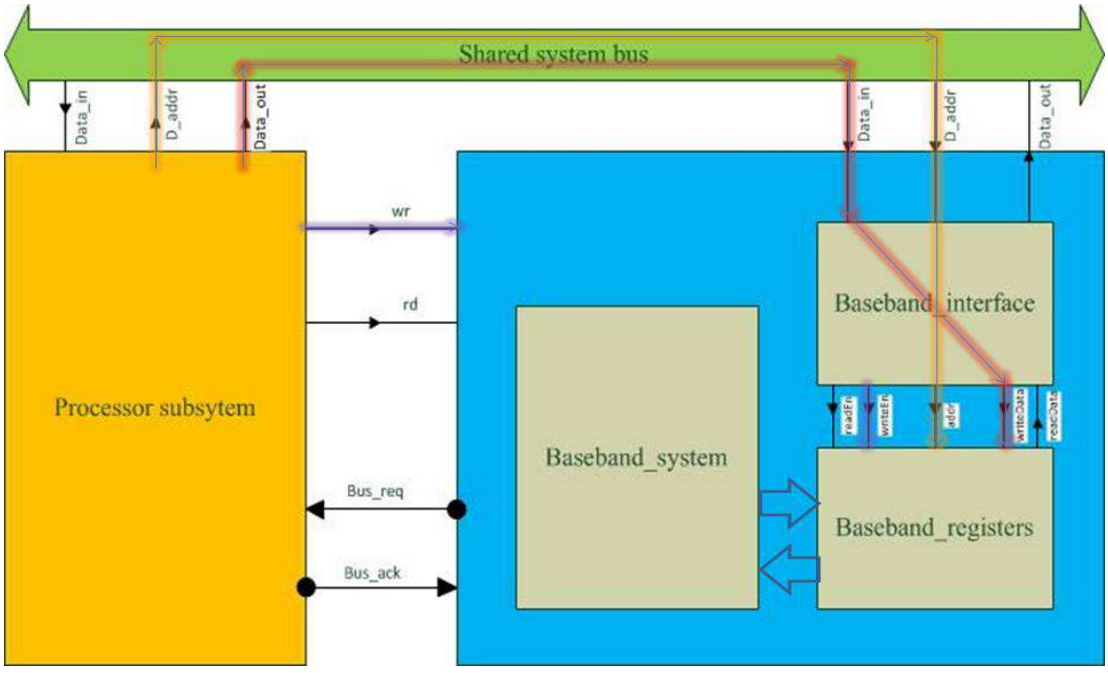

Figure 5.2 Data flow for a write operation

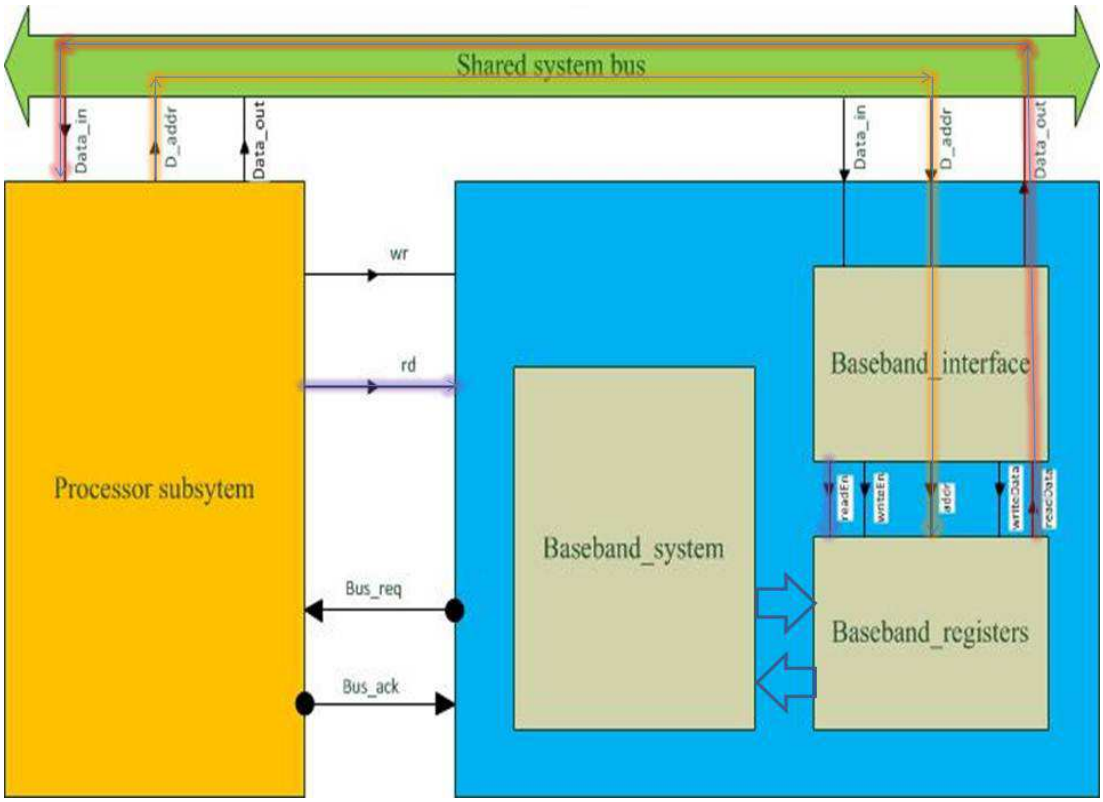

Figure 5.3 Data flow for a read operation

5.4 INTERFACE – FUNCTIONAL IMPLEMENTATION

5.4.1 Implementation of baseband_interface

The baseband_interface performs the following functions:

1. Generates readEn and writeEn for the registers depending on the ‘rd’ and ‘wr’ signals from the processor subsystem.
2. Creates the req-ack handshaking for the standard bus architecture.
3. Based on the bus status, it routes the data from/to the processor subsystem as according to the write or read operation.

The implementation has a state diagram to achieve the functions 2 and 3 as shown in Figure 5.4 below

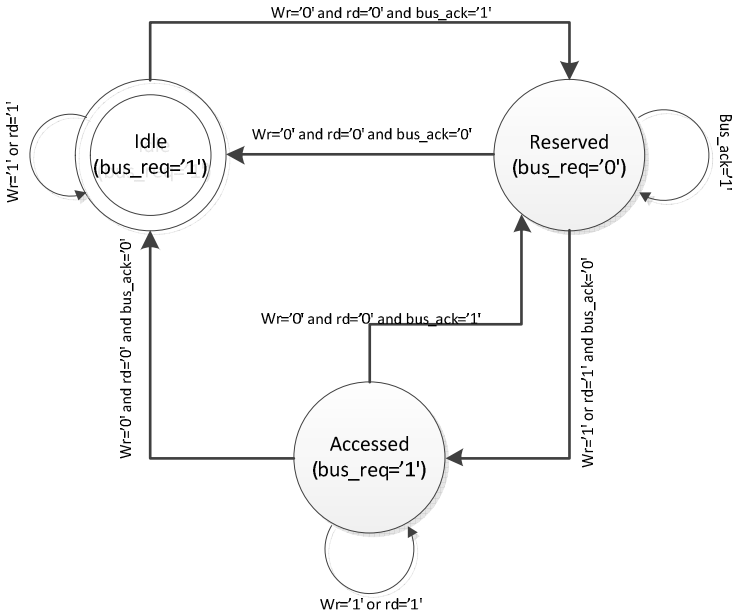

Figure 5.4 Bus state diagram

The port description of the interface is given in Table 5.1 below

Table 5.1 Port description of baseband_interface

Port Name	Direction	Description
clk_sys	In	System clock
rst_n	In	Active low reset
bus_req	Out	Bus request from baseband subsystem
bus_ack	In	Bus acknowledgement from processor subsystem
d_addr	In	Data address from processor subsystem
rd	In	Read command from processor subsystem
wr	In	Write command from processor subsystem
data_out	Out	Data to the processor subsystem for a read operation
data_in	In	Data from the processor subsystem for a write operation
regs_readEn	Out	Read enable for registers
regs_writeEn	Out	Write enable for registers
regs_readData	In	Data from registers routed to processor subsystem
regs_writeData	Out	Data from processor subsystem written to registers
regs_Addr	Out	Register address

5.4.2 Implementation of baseband_registers

The baseband_registers performs the following functions:

1. Implements 32-bit register arrays of size 6. The register array size is kept as a generic and is set to a value of 6 because the baseband_system has 6 tracking channels. The register size is set to 32-bit for convenience because processor subsystem data size is 32-bit.
2. There are 14 such register arrays, 9 for the data coming from the baseband_system and 5 for the data going out.
3. Generates write enables for each of 5 different write registers if their addresses match with the incoming address and when writeEn='1'.
4. Writes the incoming data into the registers at every rising event of the clock when the respective write enables are '1'.
5. Multiplexes the different data onto the outgoing data if the address matches with the incoming address and when readEn='1'.
6. Forwards the data from registers to ports so that they reach baseband_system and likewise forwards the data from baseband_system to the registers.

The implementation has multiplexer to achieve the function 5 as shown in Figure 5.5

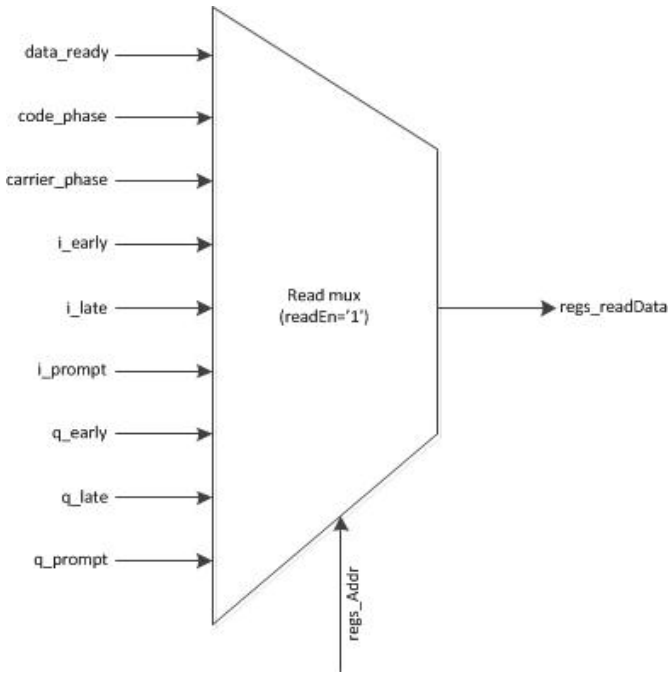

Figure 5.5 Read multiplexer for registers' readData

The port description of the registers is given in Table 5.2 below

Table 5.2 Port description of baseband registers

Port name	Direction	Description
clk_sys	In	System clock
rst_n	In	Active low reset
data_ready (6)	In	Indicates when data is ready
code_phase (6)	In	Code phase
carrier_phase(6)	In	Carrier phase
i_early(6)	In	Accumulated output of I early
i_late(6)	In	Accumulated output of I late
i_prompt(6)	In	Accumulated output of I prompt
q_early(6)	In	Accumulated output of Q early
q_late(6)	In	Accumulated output of Q late
q_prompt(6)	In	Accumulated output of Q prompt
dump(6)	Out	Signal indication for accumulation
carry_nco(6)	Out	Carrier NCO data
code_nco(6)	Out	Code NCO data
integration_time(6)	Out	Control the integration time for acquisition and tracking
sat_number(6)	Out	The satellite number controlled by software for acquisition
regs_Addr	In	Register address from processor subsystem
regs_writeEn	In	Write enable generated by baseband_interface when ‘wr’ command is 1
regs_readEn	In	Read enable generated by baseband_interface when ‘rd’ command is 1
regs_writeData	In	Write data from processor subsystem
regs_readData	Out	Read data to processor susbsystem

5.4.3 Implementation of baseband_subsystem

The baseband subsystem is just a wrapper of all the three components baseband_system, baseband_registers and baseband_interface.

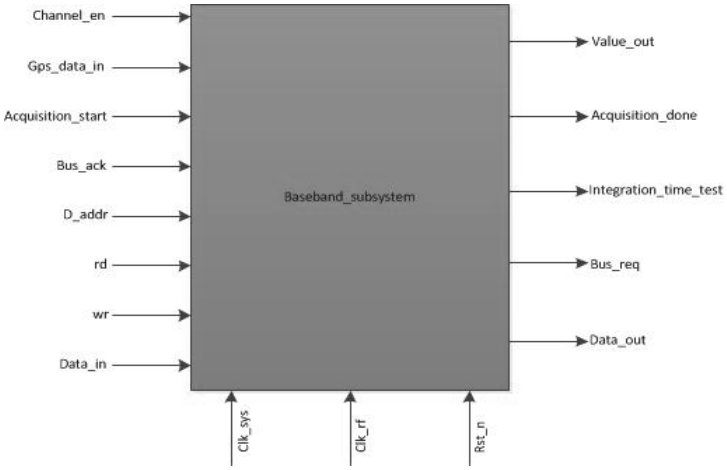

Figure 5.6 Baseband subsystem

5.5 INTERFACE – FUNCTIONAL VERIFICATION

The interface is verified by code review as a first step. This helped to immediately point out the errors in the code. Then, it was also verified by functional simulation by creating proper stimulus in the test bench and verifying the results in the waveform window. Figure 5.7 below shows that when there is no bus request and write/read controls, the data bus will be in ‘idle’ state holding the idle address and data. If there is a bus request following this, the data bus will go to ‘reserved’ state.

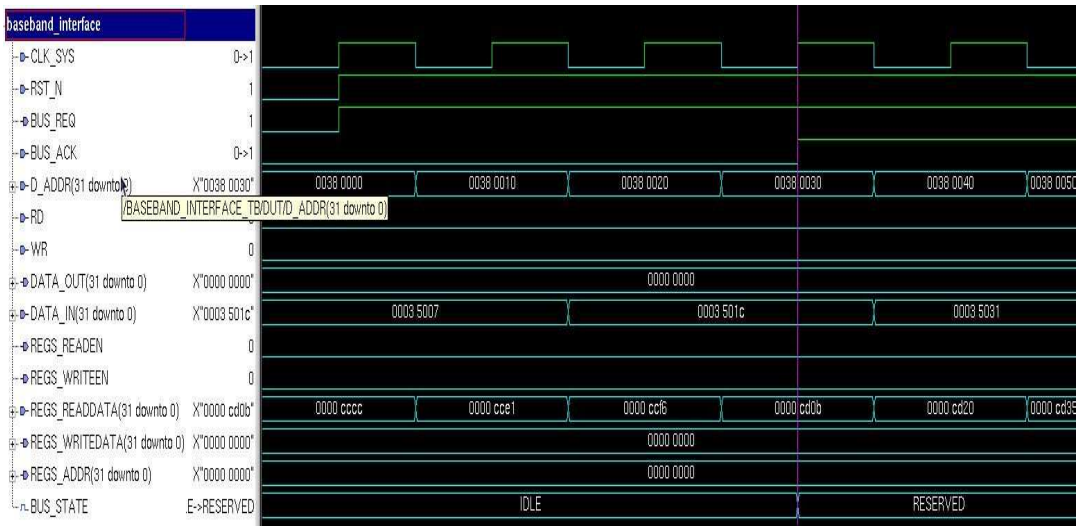

Figure 5.7 Waveforms: Idle -> Reserved

Figure 5.8 shows that as long as there is a bus request, the data bus will be in ‘reserved’ state. If there is a following write or read control signal, the data bus will go to ‘accessed’ state during which it performs the corresponding write or read access.

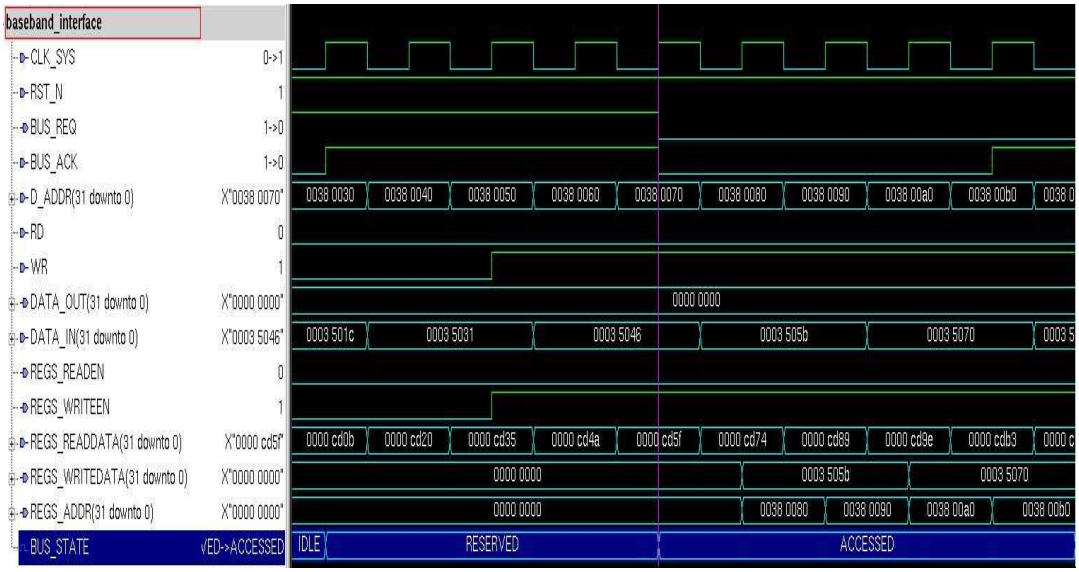

Figure 5.8 Waveforms: Reserved -> Accessed

When there is a write or read control signal, the data bus will be in ‘accessed’ state during which it performs the corresponding write or read access. If the bus request continues following this, the data bus will go to ‘reserved’ state so as to perform another write or read operation. This is shown in Figure 5.9 below.

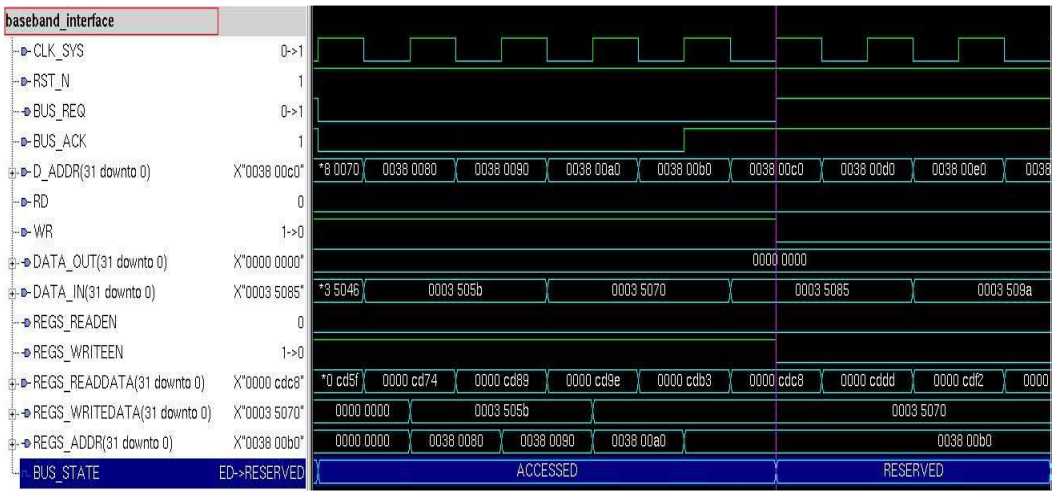

Figure 5.9 Waveforms: Accessed -> Reserved

Figure 5.10 shows that as long as there is a bus request, the data bus will be in ‘reserved’ state. If there is no write or read control signal following it, the data bus will go to ‘idle’ state during which it samples the address and data values.

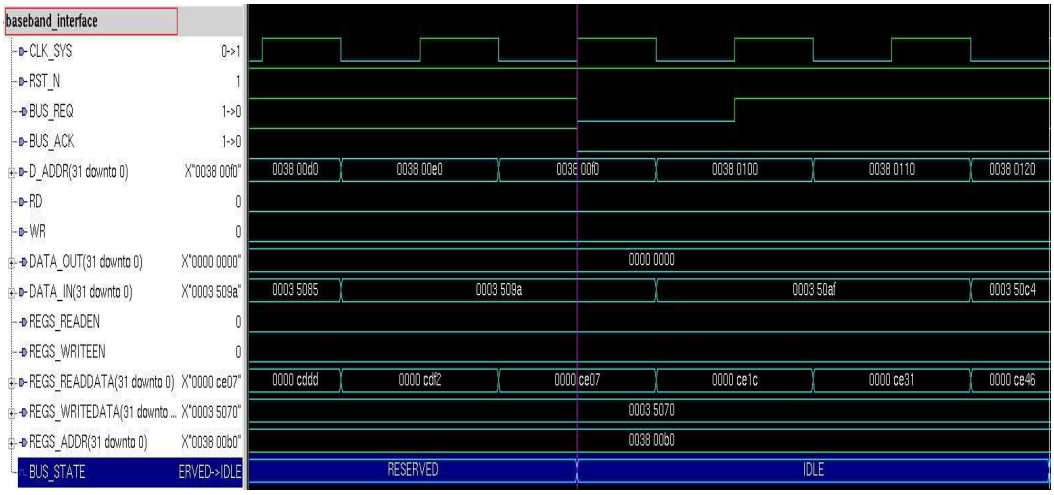

Figure 5.10 Waveforms: Reserved -> Idle

When there is a write or read control signal, the data bus will be in ‘accessed’ state during which it performs the corresponding write or read access. If there is no action following this, the data bus will go to ‘idle’ state keeping the same address and data values. This is shown in Figure 5.11 below.

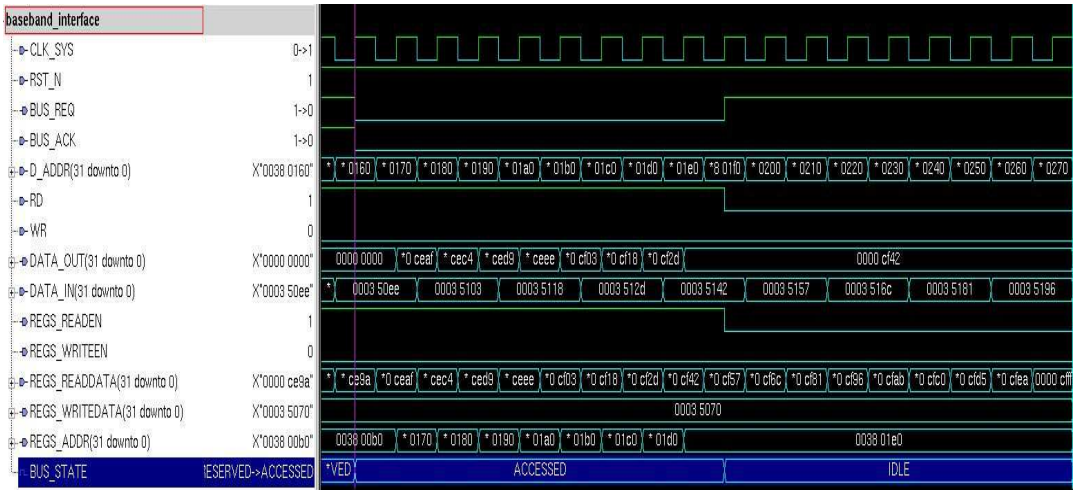

Figure 5.11 Waveforms: Accessed -> Idle

6 *Verification of Baseband Subsystem*

6.1 VERIFICATION STRATEGY

GNSS receiver reference design is verified in a dynamic simulation environment. The dynamic environment has three complementary steps: stimulus generation, response checking, and coverage analysis. ModelSim simulator is used for dynamic simulation of subsystems and their interface. Figure 6.1 shows the dynamic simulation flow of the system. TestBench (TB) creation covers the step of generating the stimuli that fully exercises the Design Under Test (DUT), improves verification coverage, exercises corner cases and finds bugs.

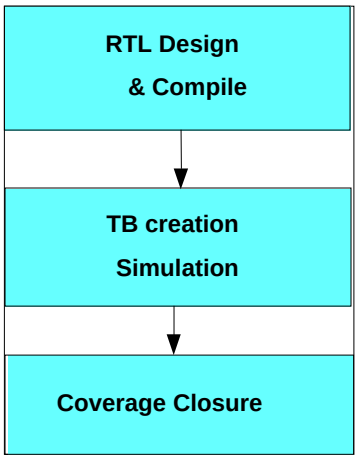

Figure 6.1 Dynamic simulation flow

6.1.1 Coverage closure strategy

Overall code coverage of each subsystem and their interface is planned to be used as a coverage closure criteria. Code coverage is an effective means of acquiring a good knowledge of low-level design details, which in turn, improves the quality of verification (Gluska, 2006). Target is to achieve maximum code coverage on BB subsystem and its interface. Coverage metrics determine the quality of verification. An uncovered metric could be the source of a potential design error. To complete the coverage, it is necessary to write more tests or to increase test stimuli. Following code coverage metrics are planned to be measured:

- Statement coverage:
  - It measures statements that are exercised by the TB stimuli during a simulation run.
  - It is a control-flow metric that tracks which statements have taken the flow of execution through the VHDL code.
  - It covers both signal and variable assignments.

- Branch coverage:
  - It measures the decision statements that affect the control flow of the code execution.
  - It covers if-then-else statements, case-statements and while-statements.

- Condition coverage:
  - It checks the truth or falsity of conditional expressions and sub-expressions.
  - It covers conditional expressions in ternary statements.

- Expression coverage:
  - It is similar to condition coverage.
  - It checks the expressions in concurrent signal assignments.

6.2 VERIFICATION PLAN

Critical applications are required to demonstrate that testing is done extensively. Hence, the plan is to create test bench for every module of the baseband subsystem to verify the Register Transfer Level (RTL) level functionality of the modules. This is shown in Figure 6.2 below.

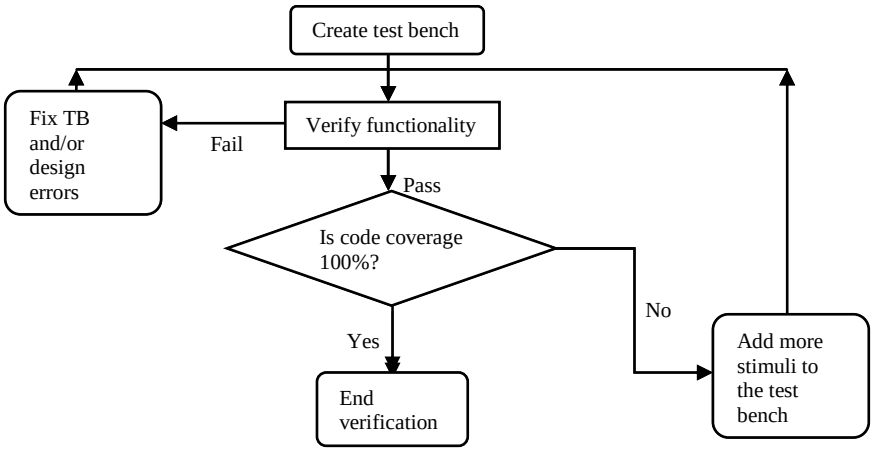

Figure 6.2 Verification plan

The coverage numbers provide a quantitative way of measuring the extent to which the verification is complete. There is not just one metric that is accepted as the only complete and reliable metric. Therefore, multiple metrics are used here to evaluate the completeness. The reports show which parts of the RTL are covered and more importantly, which are not. These numbers are then used to add more stimuli to the test bench so as to target the testing towards untested or poorly tested parts of the RTL. The development of more tests and running simulations continue until there are no holes left in the defined functional coverage plan.

Although 100% coverage cannot guarantee a 100% error-free design, it provides a more systematic way to gauge the completeness of the verification process (Chien-Nan Jimmy Liu, 2000). Once the acceptable coverage is achieved for this design, the verification can be completed.

The verification plan thus ensures both high efficiency and high reliability by using hand-written test benches and code coverage analysis as its complement.

6.3 VERIFICATION RESULTS

Each of the modules in the baseband subsystem had its own TB. These test benches were iterated to add more stimuli. After the iterations, all the coverage metrics improved. This is presented in this section for all the modules. This section also presents the simulation waveforms of some of the modules.

6.3.1 Design structure

The baseband subsystem has a structure of modules like shown in Figure 6.3 below:

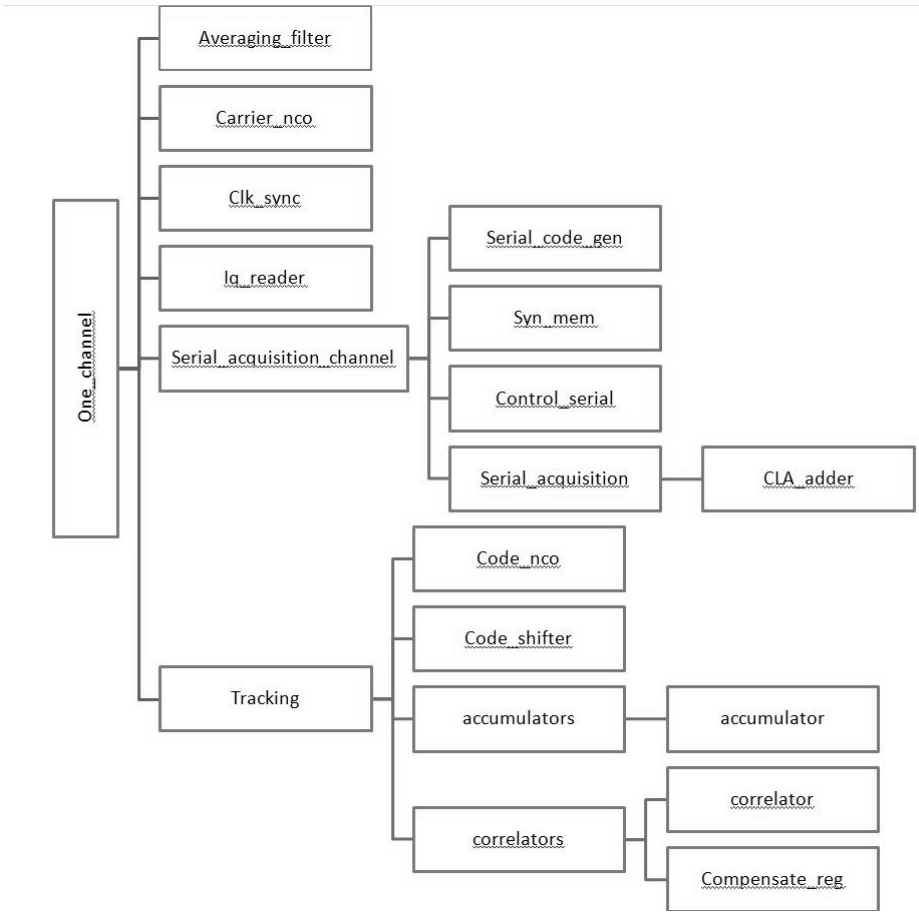

Figure 6.3 Structural hierarchy of Baseband subsystem

6.3.2 Coverage measurement tool

A fast and convenient coverage measurement tool is the simulator itself. Only one simulation run is needed for the coverage report that will include all the coverage

metrics like statement, branch, condition and expression. Figure 6.4 below shows such a simulator window for the module serial_code_gen as an example.

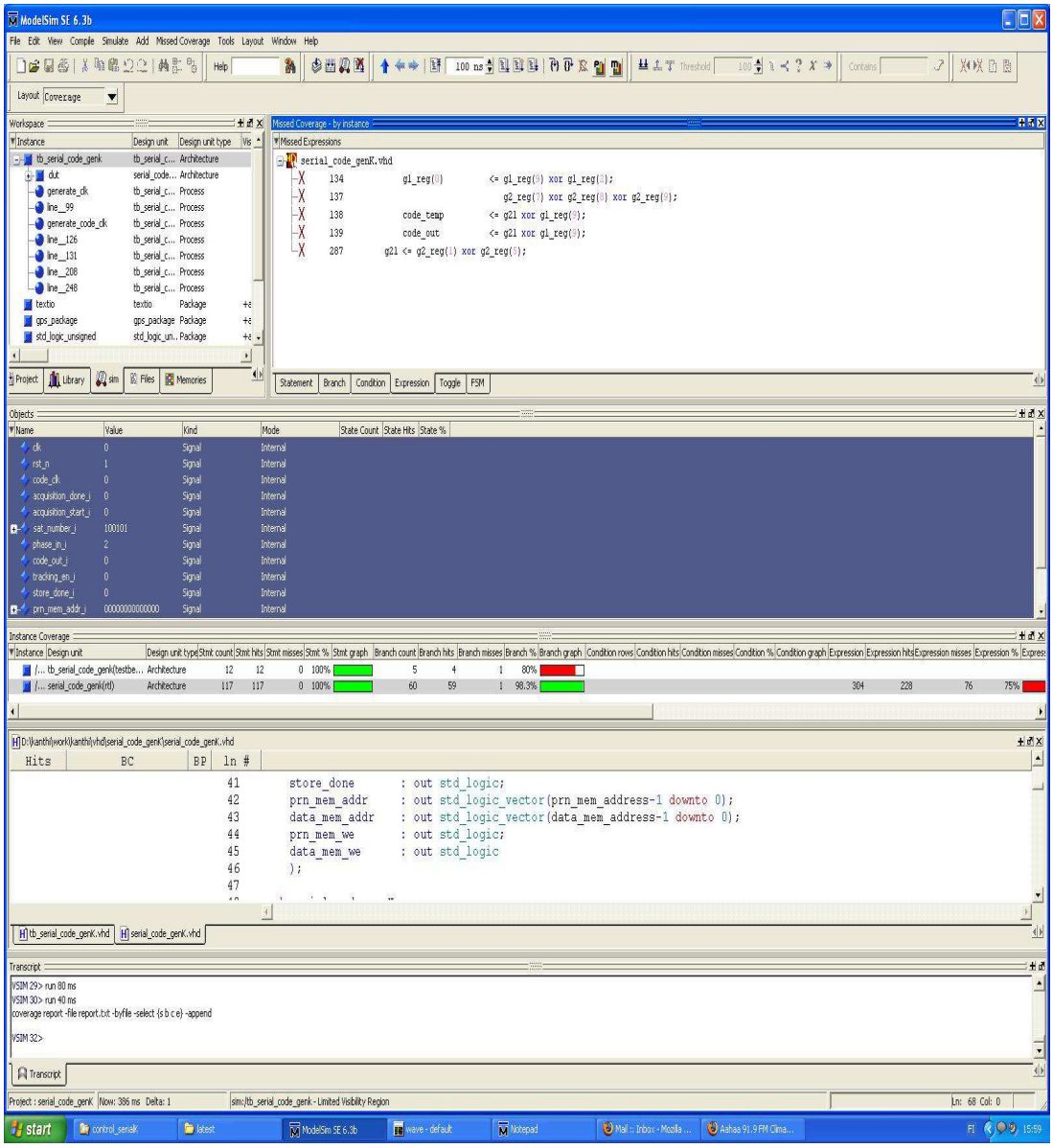

Figure 6.4 Coverage layout of ModelSim simulator

The simulator window has different panes distributed in 5 rows. The left side of the top row shows the design hierarchy. The right side pane of the same row shows the missed coverage points for all the metrics. Here, it shows the 25% of the expressions misses. The second row pane shows the signal objects. The third row pane shows the coverage report data. The fourth row pane shows the RTL code and the fifth row pane shows the transcript window.

6.3.3 Coverage results

Test bench was created for each of the modules shown above. Creating test benches for all the modules separately provided a good focus on the basic functions of each module. The coverage results of many modules were 100% in the first iteration itself. For some modules with lesser coverage percentage, more iterations of verification were done after adding more stimuli to the test bench. Such modules are shown below.

- **Serial_code_gen:**

The statement coverage has been improved from 32.5% to 100% and branch coverage has been improved from 70% to 100% whereas the expression coverage has been improved from 49% to only 75%.

Table 6.1 Coverage data for three iterations of 'serial_code_gen'

Coverage Report Summary - serial_code_gen							
Enabled Coverage	Active	Iteration 1		Iteration 2		Iteration 3	
		Hits	% Covered	Hits	% Covered	Hits	% Covered
Statements	117	38	32.5	81	69.2	117	100.0
Branches	60	42	70.0	60	100.0	60	100.0
Conditions	0	0		0		0	
Expressions	304	149	49.0	149	49.0	228	75.0

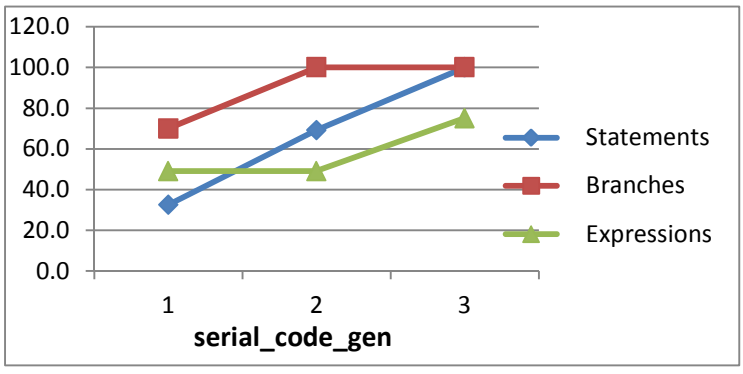

Figure 6.5 Graph showing coverage improvement of 'serial_code_gen'

- **Para_correlator:**

The statement coverage has been improved from 85.5% to 98.2% and the branch coverage has been improved from 78.9% to 84.2%

Table 6.2 Coverage data for three iterations of 'para_correlator'

Coverage Report Summary - para_correlator							
Enabled Coverage	Active	Iteration 1		Iteration 2		Iteration 3	
		Hits	% Covered	Hits	% Covered	Hits	% Covered
Statements	55	47	85.5	49	89.1	54	98.2
Branches	19	15	78.9	16	84.2	16	84.2
Conditions	0	0		0		0	
Expressions	0	0		0		0	

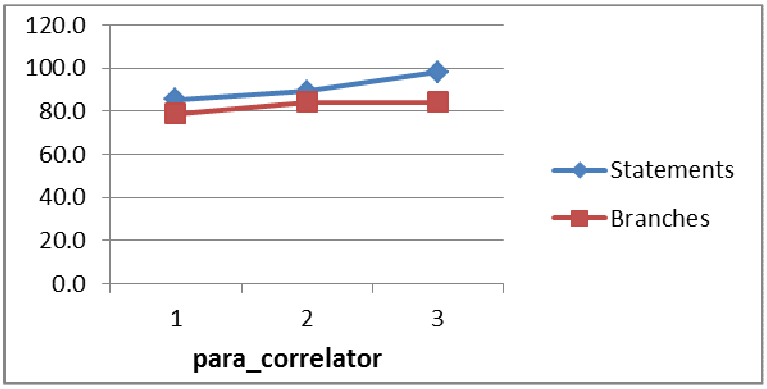

Figure 6.6 Graph showing coverage improvement of 'para_correlator'

- **Peak_detector:**

The statement coverage has been increased from 47.6% to 100%, branch coverage from 54.5% to 100% and condition coverage from 33.3% to 100%.

Table 6.3 Coverage data for four iterations of 'peak_detector'

Coverage Report Summary - peak_detector									
Enabled Coverage	Active	Iteration 1		Iteration 2		Iteration 3		Iteration 4	
		Hits	% Covered	Hits	% Covered	Hits	% Covered	Hits	% Covered
Statements	21	10	47.6	17	81.0	19	90.5	21	100.0
Branches	11	6	54.5	7	63.6	10	90.9	11	100.0
Conditions	3	1	33.3	2	66.7	3	100.0	3	100.0
Expressions	0	0		0		0		0	

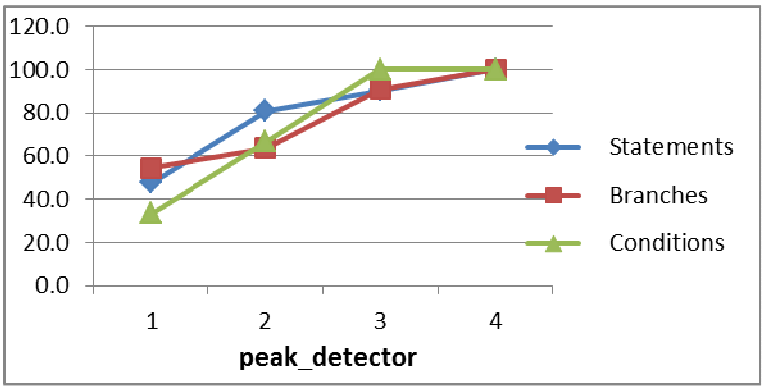

Figure 6.7 Graph showing coverage improvement of 'peak_detector'

- **Code_gen:**

The statement coverage has an improvement from 52.1% to 100%, the branch coverage from 32.1% to 98.2%, the condition coverage from 66.7% to 83.3% and the expression coverage from 27% to 100%.

Table 6.4 Coverage data for four iterations of 'code_gen'

Coverage Report Summary - code_gen									
Enabled Coverage	Active	Iteration 1		Iteration 2		Iteration 3		Iteration 4	
		Hits	% Covered	Hits	% Covered	Hits	% Covered	Hits	% Covered
Statements	94	49	52.1	84	89.4	93	98.9	94	100.0
Branches	56	18	32.1	53	94.6	54	96.4	55	98.2
Conditions	6	4	66.7	4	66.7	4	66.7	5	83.3
Expressions	300	81	27.0	228	76.0	234	78.0	300	100.0

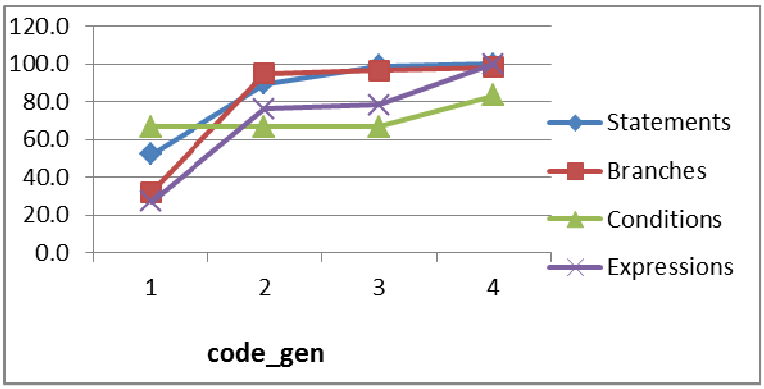

Figure 6.8 Graph showing coverage improvement of 'code_gen'

The final coverage results of the main modules of one-channel serial Baseband subsystem are shown below.

- **One_channel:** This module includes the sub-modules like averaging_filter, carrier_nco, clk_sync and iq_reader. All the sub-modules have 100% coverage in all the three coverage metrics – statement, branch and condition except for averaging_filter in condition coverage.

Table 6.5 Coverage data of all the sub-modules of 'one_channel'

Coverage Report Summary - one_channel										
Enabled Coverage	one_channel		averaging_filter		carrier_nco		clk_sync		iq_reader	
	Active	Hits	Active	Hits	Active	Hits	Active	Hits	Active	Hits
Statements	2	2	27	27	24	24	9	9	26	26
Branches	0	0	21	21	13	13	5	5	33	33
Conditions	0	0	30	20	0	0	3	3	12	12
Expressions	0	0	0	0	0	0	0	0	0	0

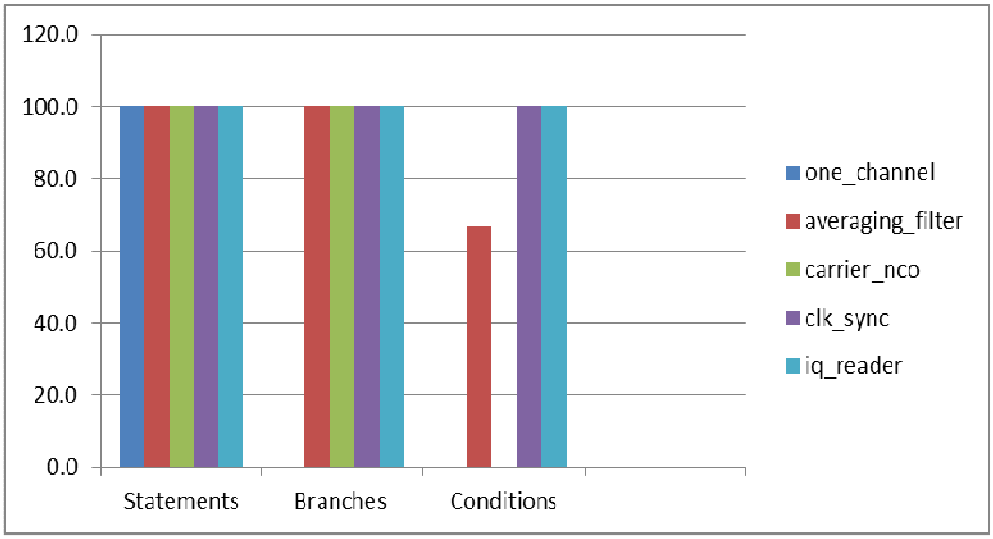

Figure 6.9 Graph showing coverage data of 'one_channel' and its sub-modules

- **Serial acq_channel**: This module includes the sub-modules like serial_code_gen, syn_mem, control_serial, serial_acq and CLA_adder. Here, serial_code_gen has only 75% expression coverage. Syn_mem has 86.7% statement coverage and 91.7% branch coverage. Control_serial has 94.1% statement coverage and 92.6% branch coverage. Serial_acq and CLA_adder has 100% coverage in all possible metrics.

Table 6.6 Coverage data of all the sub-modules of 'serial_acq_channel'

Coverage Report Summary - serial_acq_channel										
Enabled Coverage	serial_code_gen		syn_mem		control_serial		serial_acq		CLA_adder	
	Active	Hits	Active	Hits	Active	Hits	Active	Hits	Active	Hits
Statements	117	117	83	72	68	64	47	47	10	10
Branches	60	60	36	33	27	25	11	11	0	0
Conditions	0	0	0	0	0	0	6	6	0	0
Expressions	304	228	0	0	0	0	8	8	16	16

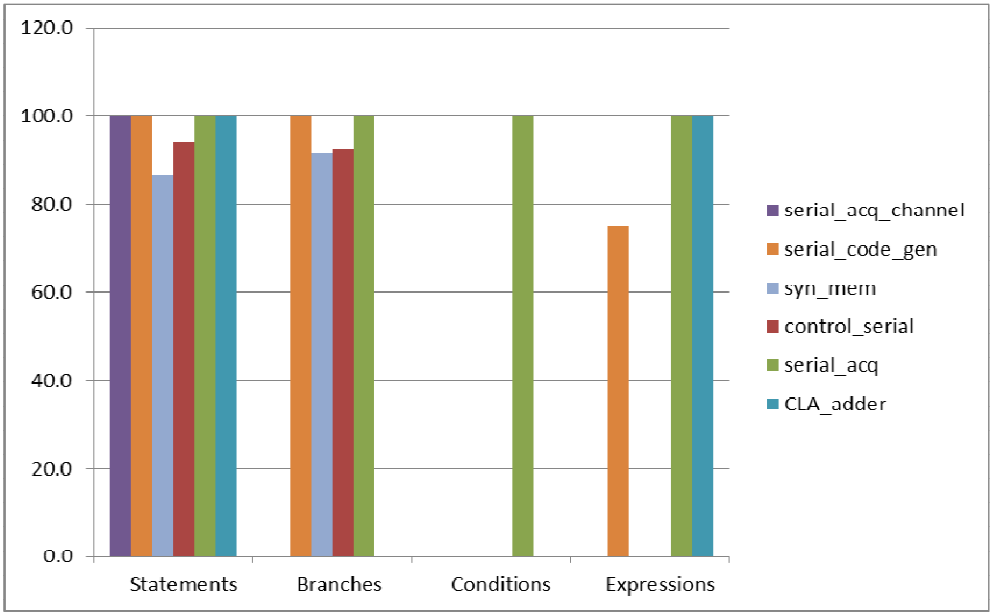

Figure 6.10 Graph showing coverage data of 'serial_acq_channel' and its sub-modules

- **Tracking**: This module includes the sub-modules like code_nco, code_shifter, accumulator, correlator and compensate_reg. All the sub-modules have the 100% coverage in all possible metrics.

Table 6.7 Coverage data of all the sub-modules of 'tracking'

Coverage Report Summary Data - tracking										
Enabled Coverage	code_nco		code_shifter		accumulator		correlator		compensate_reg	
	Active	Hits	Active	Hits	Active	Hits	Active	Hits	Active	Hits
Statements	13	13	7	7	6	6	4	4	4	4
Branches	3	3	5	5	7	7	9	9	5	5
Conditions	0	0	0	0	0	0	0	0	0	0
Expressions	0	0	0	0	0	0	0	0	0	0

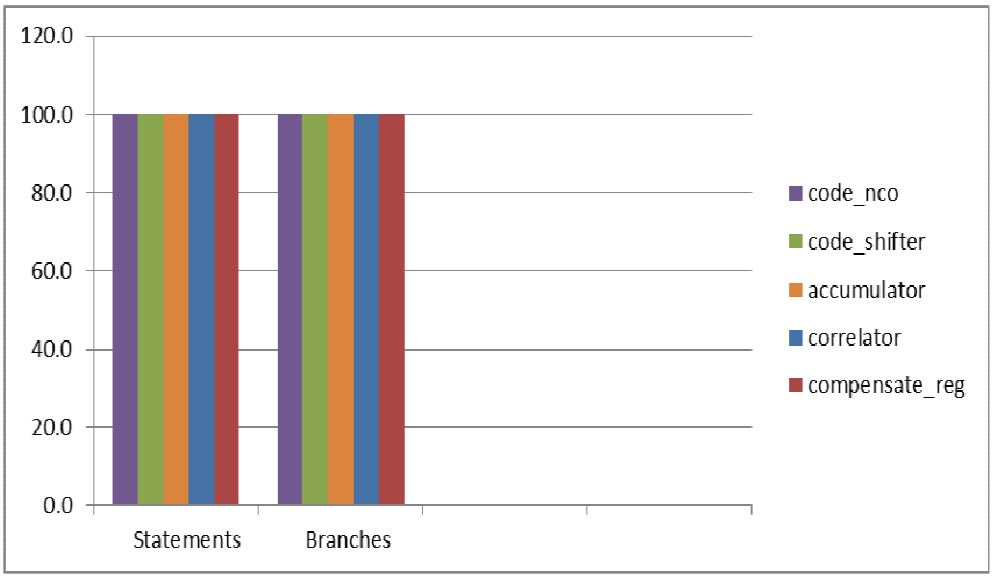

Figure 6.11 Graph showing coverage data of 'tracking' and its sub-modules

6.3.4 Simulation result

Simulation waveform of the acquisition block is shown in Figure 6.12 below. The acquisition produces the correlation result of all the PRN phases and Doppler bins for one satellite.

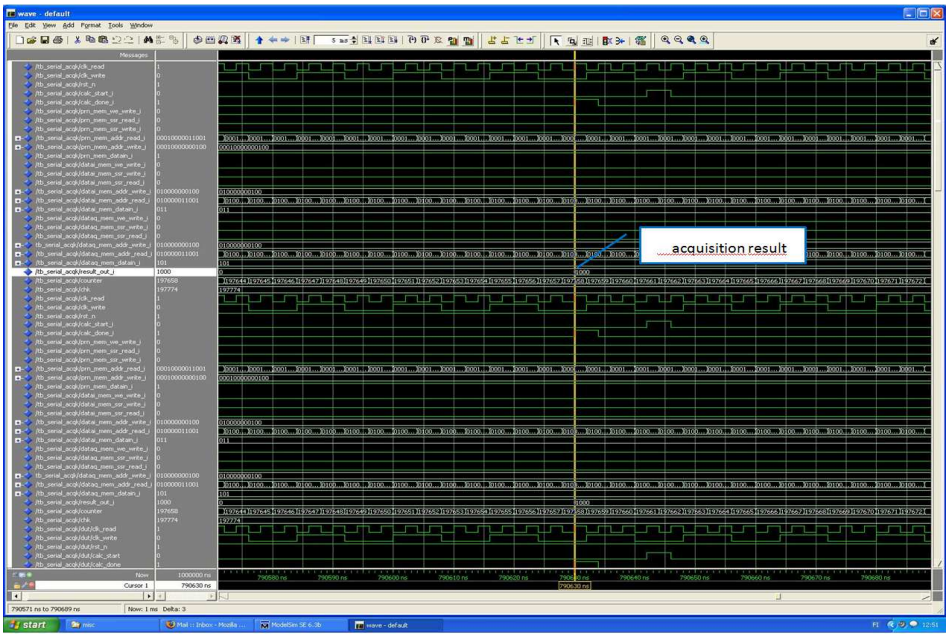

Figure 6.122 Waveform showing the simulation results of 'serial_acquisition'

7 *Synthesis*

7.1 PROCESSOR SUBSYSTEM

The COFFEE core which is a published design was previously synthesized on Altera Stratix II EP2S180 DSP development board. It hosts the device EP2S180F1020C3 which is a 1020-pin package. The kit facilitates the entire design process from design conception through hardware implementation. Some of the features of this device are (Stratix II EP2S180 DSP Development Board, Refence Manual, 2005):

- Adaptive Logic Modules: 71760, maximizes performance
- Adaptive look-up tables: 143520
- Logic array blocks: 8970
- Logic elements: 179400
- M512 RAM blocks: 930
- M4K RAM blocks: 768
- M-RAM blocks: 9
- Total RAM bits: 9383040, for demanding and memory intensive applications
- DSP blocks: 96, for efficient implementation of high performance filters and other DSP functions
- 18-bit x 18-bit multipliers: 384
- A/D converters: 2, D/A converters: 2
- I/O: 742
- Speed grade: -3, -4, -5

7.2 BASEBAND SUBSYSTEM

The baseband_system design was previously synthesized on Xilinx Virtex II Pro XC2VP30. The special features are (Xilinx Virtex-II Pro Platform FPGAs: Introduction and Overview, 2011):

- High-Performance: 8 multi-gigabit transceivers, 2 RISC processor blocks
- Flexible logic resources, logic cells: 30816
- Slices: 13696, Max. Distr. RAM: 428 Kb
- 18 x 18 bit multiplier blocks: 136
- 18 Kb blocks (SelectRAM): 136, Max. block RAM: 2448 Kb
- I/O: 644
- High-performance clock management circuitry
- Speed grade: -7, -6 and -5 with -7 having the highest performance

7.3 SYNTHESIS RESULTS

The different subsystems were previously synthesized on different FPGA platforms. For this work, they have to be synthesized on a single platform. Here, the decision is to synthesize both the subsystems on Xilinx Virtex-II Pro platform. One of the main reasons is that baseband subsystem includes many of Xilinx based models. The other reasons are below:

- Though the processor subsystem was synthesized on Altera Stratix II platform, it is also synthesizable on Xilinx Virtex II Pro platform.
- Xilinx Virtex-II Pro provides high computational capacities of up to several million logic element equivalents.
- They provide on-chip memories and DSP capabilities.
- They are very well suited for complete System-on-Chip solutions.
- It could connect to RF front-end as well as different communication interfaces.
- Thus, it has the maximum flexibility in the design and at the same time promises reductions in receiver development costs.

The target device is Xilinx Virtex-II Pro xc2vp30-6ff896. Table 7.1 below shows the logic utilization of both.

Table 7.1 Synthesis – COFFEE and baseband

Logic Utilization	Used			Available	% Utilization		
	COFFEE	Baseband	Total		COFFEE	Baseband	Total
Number of Slices	10974	462	11436	13696	80.13	3.37	83.50
Number of Slice Filp Flops	5586	417	6003	27392	20.39	1.52	21.92
Number of 4 input LUTs	18967	863	19830	27392	69.24	3.15	72.39
Number of bonded IOBs	485		485	556	87.23	0.00	87.23
Number of MULT18X18s	16	4	20	136	11.76	2.94	14.71
Number of GCLKs	1	2	3	16	6.25	12.50	18.75

8 *Conclusion & Future Work*

The first objective of this thesis was to integrate COFFEE RISC core with a 6-channel GPS L1 baseband system. Firstly, the motivation of this thesis was well understood. Secondly, the basic information and the background knowledge were obtained. Finally, the integration was accomplished by an efficient interface design. This interface is shared by both the baseband and the processor subsystems. This interface is based on standard bus architecture. The main advantages of this architecture are it is simple, economical and greater performance. The RTL implementation of this interface consists of two components – interface control logic and register database. The interface was verified functionally by creating required stimulus in the test bench. The simulation results show different behaviors of the interface.

The other objective of this thesis was to verify the baseband system exhaustively. The verification was better guaranteed by having a coverage closure strategy and improving the quality of verification. As with any design, this verification took more time as it needed 3-4 iterations to achieve a code coverage result close to 100%. The coverage results presented here prove that the targets are well achieved. Finally, FPGA synthesis tasks were performed. The baseband subsystem was synthesized on XILINX Virtex-II Pro platform as it contained some Xilinx based models. The interchangeability of synthesis platform for COFFEE core was tried and tested on the same Xilinx platform. The synthesis results show the logic utilization of both the subsystems.

The future work includes

1. Verifying the baseband processor system
    a. Developing system software
    b. Realizing both HW + SW on the FPGA platform
2. Using a baseband system that is based on a different acquisition algorithm with the same interface design.
3. Using a baseband system that could also process Galileo and other GNSS signals.

References

(2005). *Stratix II EP2S180 DSP Development Board, Refence Manual.* San Jose, CA: ALTERA.

(2011). *Xilinx Virtex-II Pro Platform FPGAs: Introduction and Overview.* Xilinx.

China Adds Details to Compass Signal Plans. (2008, September/October). Retrieved from InsideGNSS: http://www.insidegnss.com/node/803

COFFEE Core User Manual. (2007, July). Tampere: TUT.

Galileo specifications. (2007, August 16). Retrieved from ESA: http://www.esa.int/esaNA/SEM86CSMD6E_galileo_0.html

Berger, G.L. & Goiser, A.M.J. (1996). Synchronizing a Digital GPS Receiver. *MELECON '96.* Bari, Italy.

Bergeron, J. (2000). *Writing Testbenches (functional verification of HDL models).* New York: Kluwer Academic Publishers.

Braasch, M.S. & van Dierendonck, A. J. (1999). GPS Receiver Architectures and Measurements. *Proceedings of the IEEE* (Volume:87, Issue:1, pp.48 - 64). IEEE.

Chong, C. (2009). Status of Compass/BeiDou Development. *Stanford's 2009 PNT Challenges and Opportunities Symposium.* Stanford.

Diggelen, F. v. (2009). *A-GPS: Assisted GPS, GNSS, and SBAS.* U.S.A.: Artech House.

Garzia, F. (2005). *Design of a peripheral set for a RISC microprocessor*. Tampere: TUT.

Gluska, A. (2006). Practical methods in coverage-oriented verification of the merom microprocessor. *Design Automation Conference, 43rd ACM/IEEE* (pp. 332-337). SFO, CA, USA: DAC '06.

Grace XingXin Gao, Alan Chen, Sherman Lo, David De Lorenzo, Per Enge. (2007, July/August). GNSS over China, The Compass MEO Satellite Codes. *InsideGNSS*, pp. 36-43.

Guenter W. Hein, et al. (2002). Status of Galileo Frequency and Signal Design. *Members of the Galileo Signal Task Force of the European Commission.* Brussels.

Hurskainen H., et al. (2008). GNSS Receiver Reference Design. *ASMS 2008* (pp. 204 - 209). Bologna: ASMS.

Hurskainen H., et al. (2009). Multicore Software-Defined Radio Architecture for GNSS Receiver Signal Processing. *EURASIP Journal on Embedded Systems 2009*.

Kaplan, E.D. & Hegarty, C.J. (2006). *Understanding GPS: Principles and Applications.* Norwood, MA 02062: Artech House, Inc.

Kylliainen J., et al. (2003). COFFEE - A core for free. *International Symposium on System-on-Chip,* (pp. 17 - 22). IEEE.

Lehto, S. (2005). *Architecture Exploration for Enhanced GPS and Galileo Tracking.* Tampere: M.Sc. Thesis, TUT.

Liu, C.-N.J., Chen-Yi Chang, Jing-Yang Jou, Ming-Chih Lai, Hsing-Ming Juan (2000). *A Novel Approach for Functional Coverage Measurement in HDL*. Geneva, Switzerland: ISCAS.

Liu, Z. (2008). *An Advanced Way to Implement Acquisition Algorithm in GNSS Receiver Design.* Tampere: M.Sc. Thesis, TUT.

Manuel Perez-Ruiz & Shrini K. Upadhyaya. (2012). *GNSS in Precision Agricultural Operations.* INTECH.

Mendizabal, J., et al. (2009). *GPS & Galileo: Dual RF Front-end Receiver and Design, Fabrication, and Test.* The McGraw-Hill. p. 159

Michael J. Flynn & Wayne Luk. (2011). *Computer System Design: System-on-Chip.* New Jersey: John Wiley & Sons, Inc.

Nieminen, L. (2007). *Implementation of GNSS Baseband Hardware.* Tampere: M.Sc. Thesis, TUT.

Plausinaitis, D. (2009). GPS Signal Acquisition. *Danish GPS Center.* http://kom.aau.dk/~dpl/courses/mm11_slides.pdf

Revnivykh, S. G. (2009). GLONASS Program Update. *4th Meeting of International Committee on GNSS.* St.Petersburg: Federal Space Agency, The Russian Federation. http://www.oosa.unvienna.org/pdf/icg/2009/icg-4/01.pdf

Thomas Grelier, et al. (2007, May/June). Initial Observations and Analysis of Compass MEO Satellite Signals. *InsideGNSS*, p. 39-43.

Tsui, J. B.-Y. (2000). *Fundamentals of Global Positioning System Receivers, A Software Approach.* New York: John Wiley & Sons, Inc.